The Wunderkammer Interface

A DISSERTATION SUBMITTED TO THE FACULTY OF
UNIVERSITY OF MINNESOTA
BY

Benjamin J M Klein

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Dr. Alex Lubet, Advisor

May 2017

Acknowledgements

Thank you to the teachers at the University of Minnesota with whom I had the privilege to work. This dissertation would not be possible without your knowledge and support.

Special thanks to Yan Pang, Jay Afrisando, Benjamin Cold, Tyler Cessnor, Ron Coulter, Marcelo Rilla, Samu Gryllus, Adam Conrad, at the musicians of the St. Paul Conservatory. It was a pleasure to collaborate with you on this project.

Dedication

To Ruth and Jocie for their loving support

Abstract

The Wunderkammer Interface is a nonhierarchical composition that invites an improvising musician to interact with twenty-five modules of composed digital audio processing. The musician who is engaging with the interface has the option to activate these modules independently, as a progression, or as overlapping processing schemes. *The Wunderkammer Interface* was designed in the SuperCollider audio processing environment.

The interactive structure of *The Wunderkammer Interface* is based on the fluidity between the roles of creator, participant, and listener possible through this work. The performances described in this paper demonstrate unique realizations of *The Wunderkammer Interface* through different interactions with the interface.

Supplementary Audio Tracks:

Track 1 - demonstration of individual modules, wind chimes
Track 2 - demonstration of Series 1, wind chimes
Track 3 - demonstration of Series 2, wind chimes
Track 4 - demonstration of Series 3, wind chimes
Track 5 - demonstration of Series 4, wind chimes
Track 6 - improvised module activation, wind chimes
Track 7 - Yan Pang, piano
Track 8 - Yan Pang, piano
Track 9 - Yan Pang, piano
Track 10 - Yan Pang, piano
Track 11 - Yan Pang, piano
Track 12 - Jay Afrisando, piri with a saxophone mouthpiece and saluang
Track 13 - Jay Afrisando, piri with a saxophone mouthpiece and saluang
Track 14 - Benjamin Cold, alto saxophone
Track 15 - Benjamin Cold, alto saxophone
Track 16 - Benjamin Cold, alto saxophone
Track 17 - Benjamin Cold, alto saxophone
Track 18 - Benjamin Cold, soprano saxophone
Track 19 - Benjamin Cold, soprano saxophone
Track 20 - Ron Coulter, elephant bells
Track 21 - Ron Coulter, audubon bird call, turkey call, clay ocarina
Track 22 - Ron Coulter, tam-tam, various implements
Track 23 - Ron Coulter, drum set
Track 24 - Marcelo Rilla, spoken text
Track 25 - Marcelo Rilla, electronics
Track 26 - Samu Gryllus, misc. household objects, vocalization
Track 27 - Samu Gryllus, environmental sound
Track 28 - Samu Gryllus, clapping, snapping, vocalization

Table of Contents

List of Figures

Chapter 1: Wunderkammern

The Wunderkammer, or cabinet of wonder, is a creative trope derived from collections that invite exploration. These collections range from personal hoards and museum exhibits to encyclopedic collections of ideas or shared knowledge available through electronic networks. The Wunderkammer, as a creative trope, was the point of departure for the development of *The Wunderkammer Interface*. This project invites the exploration of a collection of live processing schemes, programmed in the SuperCollider digital processing environment, through improvisatory engagement with the interface. The music realized from this project is the result of the combination of the design embedded in digital processing schemes and the voice of the performer who engages with the work. The documented recordings of this project demonstrate the contrast of musical expression that can be derived from it. I present this growing collection of musical events as a Wunderkammer of sound.

The early Wunderkammer, or Kunstkammer, refers to the collections amassed by the nobility of the sixteenth and seventeenth centuries that were meant to display the collector's impressive comprehension of world as they knew it. In an era of expanding boundaries through empire and exploration, artifacts retrieved from these ventures represented for rulers like the Habsburgs of the Holy Roman Empire the accumulation of their knowledge in parallel with the expansion of territory through their conquests.[1] As the Baroque era gave way to the Age of Enlightenment, the act of amassing similar collections became a practice for a larger educated upper class in both Europe and the

[1] Werner Muensterberger, *Collecting, an Unruly Passion* (Princeton: Princeton University Press, 1994), 189-200.

United States. Thomas Jefferson, for example, maintained a collection of Native American artifacts, art objects, and natural specimens in the entranceway at Monticello.[2] Many of these Wunderkammern, Kunstkammern, and cabinets of curiosity became the source collections for the first museums.

While artists like Domenico Remps depicted these collections through their art in the sixteenth and seventeenth centuries[3], the concept of the Wunderkammer as a work of art itself is relatively new. Marcel Duchamp's *Box in a Valise (From or by Marcel Duchamp or Rrose Sélavy),* for example, is a series of reproductions of his own work that Duchamp packed into transportable carrying cases produced between 1935-1941, each a complete kit of his work up until that point.[4] Contemporary artists like Damien Hirst refer to these Wunderkammern more directly. In his work *Forms without Life* created in 1991, Hirst displays a collection of shells arranged like scientific specimens.[5]

The parallels between the Wunderkammer collector's pursuit to amass their personal manifestation of the universe and the pursuit of early encyclopedists in writing an all-encompassing tome of knowledge offers an interpretation of the Wunderkammer as a collection of ideas. Guerino Mazzola describes the initial endeavors of developing an encyclopedia as the formalizing and categorizing of Aristotelian dialectics.

> "[A spatial understanding] of concepts is applied not only in abstract but also in artistic operationality in the Encyclopaedia of the Baroque. The field of all possible knowledge of the Ars Magna of a Raymundus Lullus provides a topology of the all knowable and this in Johann

---

[2] Joyce Henri Robinson, "An American Cabinet of Curiosities," in *Acts of Possession, Collecting in America*, ed. Leah Dilworth (New Brunswick: Rutgers University Press, 2003), 19.

[3] Richard Gregory, "Trumping Eyes, Part 2," *Perception*, 38 (2009): 1265-1266, DOI:10.1068/p3809ed, 1256.

[4] Brenna Campbell, Élodie Lévèque & Erin Jue. "Marcel Duchamp's Boîtes-en-valise: Collaboration and conservation." *Studies in Conservation,* 57 (2012): sup1, S52-S60, DOI: 10.1179/2047058412Y.0000000017

[5] Ann Gallagher, "Introduction," in *Damien Hirst*, ed. Ann Gallagher, 11-19. (London: Tate Publishing, 2012), 17.

> Heinrich Alsted's post-creative formulation in concrete terms as an alphabetically coordinated three-dimensional field of possibilities."[6]

A more focused exercise in a taxonomy of knowledge is Max Brückner's collection of polyhedra theorized and depicted in *Vieleck un Vielflache: Theorie und Geschichte* published in 1900.[7] He demonstrates the boundless variation that can be achieved in the conceptualization of a three dimensional shape, creating for himself a universe of forms. The numerous pages of polygons that he provides in this book resemble the shelves of artifacts that existed in the Wunderkammern of previous centuries.

In the conclusion of his book, *The Lure of Antiquity and the Cult of the Machine: The Kunstkammer and the Evolution of Nature, Art and Technology*, Horst Bredekamp alludes to the parallels between the early Kunstkammer, and the amalgamation of image-based content accessible through the Internet.

> "No one wants to return to the deliberate chaos of the Kunstkammer as museums. But the boundaries between art, technology, and science are beginning to break down in a similar manner as has been demonstrated by the Kunstkammer. In view of this fact, their lessons of visual association and thought processes which precede language systems take on a significance which might even surpass their original status. Highly technological societies are experiencing a phase of Copernican change from the dominance of language to the hegemony of images."[8]

The chaos that Bredekamp wants to avoid is a consequence of the multitude of contributing voices that make up the body of content available on the World Wide Web. This chaos of connections, however, has fostered many inventive and creative artworks,

---

[6] Guerino Mazzola, "EncycloSpace - the Knowledge Space in the Information Age," trans. Benjamin Klein, *Humanities @ Encyclospace,* access date April 6, 2017, http://www.encyclospace.org/special/encyclospace-CD/html/chapter11/main.html

[7] Bruckner, Max. *Vielecke und Vielflache: Theorie und Geschichte* (Leipzig: B. G. Treubier, 1900).

[8] Horst Bredekamp, *The Lure of Antiquity and the Cult of the Machine: The Kunstkammer and the Evolution of Nature, Art, and Technology*, trans. Allison Brown. (Princeton: Marcus Wiener Publishers, 1995), 113.

solutions to social and political dilemmas, and scientific breakthrough through the collaborative possibilities available through it.

I used the Wunderkammer examples presented here as the conceptual models on which to base my project. *The Wunderkammer Interface* is a collection of twenty-five modules (Fig. 1), each of which digitally process an incoming audio signal in a unique way. The musician who is engaging with the interface has the option to activate these modules independently, as a progression, or as overlapping processing schemes. I present a critical mass of processing options to a musician who engages with *The Wunderkammer Interface* so that the entirety of the possible combinations, or manifestations, of the project cannot be exhausted in one, or a few, engagements.

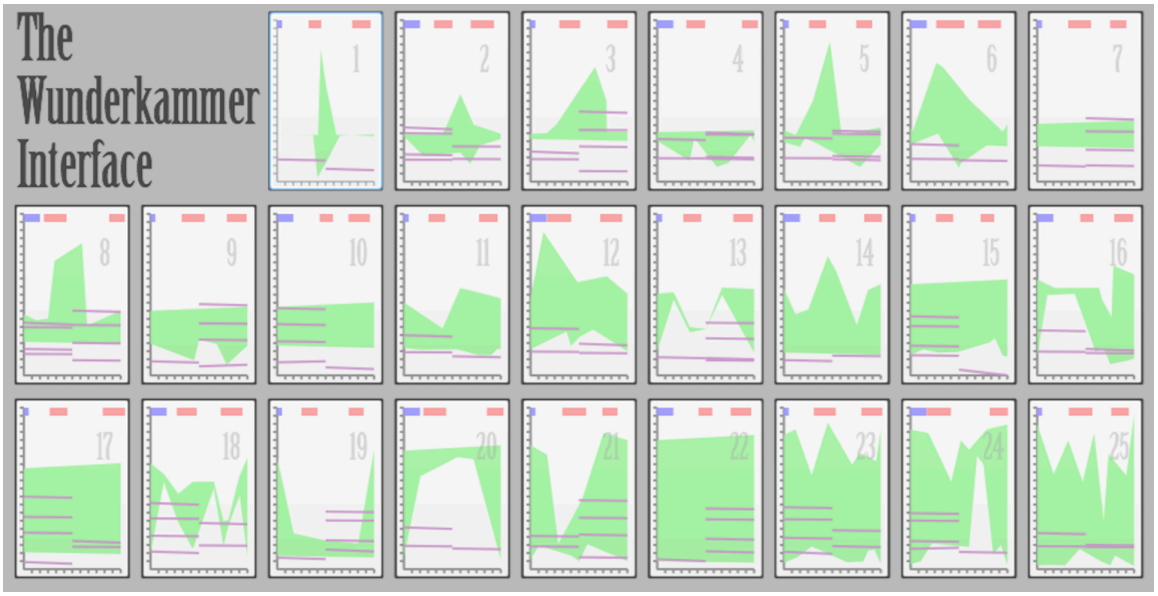

Figure 1 - *The Wunderkammer Interface*, module detail

I used Wunderkammer curatorial strategies as compositional techniques. In the early Wunderkammer, I recognize that each object included in the collection is valued for

its own exquisite nature. While at the same time, this object does not necessarily need to be complex. I designed the modules of *The Wunderkammer* based on this exquisite simplicity. Each module of *The Wunderkammer Interface* triggers a processing scheme that incorporates gestural nuances within underlying formal progressions, while limiting the amount of processing in each module to only a few parameters.

The complexity of *The Wunderkammer Interface* lies in the combinatory possibilities that exist in the modules as a collection. The manifestation of the possible forms in Max Brückner's collection of polyhedrons, for example, unpacks the experience of accumulative knowledge. In my project, I design my modules so that they encourage those who engage with it to draw multiple connections between modules, developing an increasingly complex awareness of its structure. I also rely on the unpredictable blooming of variation that results from the contribution of other voices from collaborating musicians as an essential source of this work's aesthetic appeal.

I also recognize that a Wunderkammer can act as a vehicle for reckless appropriation. The cultural significance of the artifacts from indigenous communities found in Jefferson's Monticello collection, for example, is lost in the alien environment and inaccessibility of Jefferson's foyer. The lack of attribution of authorship further alienates the creators of these objects from their work. In a significant portion of his artwork, Damien Hirst incorporates the bodies of dead animals, drawing a morbid curiosity that, as Brian Dillon points out in his essay on Hirst's work, reminds the viewer of the practice of anatomy as public spectacle exercised for centuries.[9] My intent is not to use *The Wunderkammer Interface* as a vehicle of appropriation, but as a mechanism of

[9] Brian Dillon, "Ugly Feelings," in *Damien Hirst*, ed. Ann Gallagher, 21-29. (London: Tate Publishing, 2012), 23.

collaboration. I am, however, interested in the dilemmas that an interactive composition like, *The Wunderkammer Interface*, pose regarding authorship and appropriation. Is it a composition, a guided improvisation, or a digital instrument? There is a different distribution of creative credit indicated by each of these scenarios. *The Wunderkammer Interface* draws attention to this dilemma as do the socio-politically charged collections of wonders presented here.

Chapter 2: Modes of Interaction in Electroacoustic Music

Composition projects that fit within the idiom of electronic music, and a continuing participation in various improvisation communities, are two focuses of creativity in which I have oriented my work over the past decade. I often combine these two streams of creativity. I have had the opportunity to perform in a variety of musical contexts presenting original work for tuba and live electronics, works that act as frameworks for extended solo improvisation.[10] *The Wunderkammer Interface* is a project in which I move away from realizing my work as a performer/composer to further explore the aesthetic appeal associated with an improvisor's use of digital live processing elements in the context of performance.

In my own practice, I find that the designations between composer, performer, and audience often become blurred because the incorporation of digital technology as a means to perform a work, or the use of technology to realize an improvised event, invites different interactions between these groups than what is assumed in the traditional concert hall. *The Wunderkammer Interface* is a project that invites the investigation of these blurred definitions through interactivity. I use the terms creators, participants, and listeners as useful designations to describe the interactions that occur in my own work and in the works of others. These designations infer a more fluid exchange between and merging of each role. I use the following definitions of these identities as a way to compare the interactions between those engaged with electroacoustic musical works. A creator is anyone who contributes to forming the structure or idea of a work at the point

[10] Benjamin J Mansavage Klein, "Speech, Rotation, Tuba: a Progression of Works, 2008 - 2010" (master's thesis, Wesleyan University, 2010).

of performance or previous to the music's realization. A participant is anyone who consciously engages in making sound to realize the music. A listener is anyone who lends their attention to hearing the music realized. Designations of identities are a tool used by other musicians in practice and scholarship to frame interactivity conducive to their respective works as well. Anthony Braxton, for example, invites those who wish to engage in his music as friendly experiencers, a designation that includes both those listening to the music and those performing it.[11]

The mode of interaction between actors in a network is highly consequential to the observations that can be made of that network, as demonstrated by the research of Insuk Lee, Eiru Kim, and Edward M. Marcotte.[12] I examined six works that demonstrate different modes of interaction between the roles of creator, participant and listener. I chose these six works because I identify the sonic realizations of these compositions as dependant upon the interaction that occurs between different actors who engage with the music. The works I compared are: *Voyager* (1993) by George Lewis, *I am Sitting in a Room* (1969) by Alvin Lucier, *Rainforest* (1966-1973) by David Tudor, *Solo für Melodieinstrument mit Rückkopplung* (1966) by Karlheinz Stockhausen, *Modular Live Interface* (2012) by Sam Pluta, and *#[unassigned]* (2003) by James Saunders.

Voyager **(1993) by George Lewis**

George Lewis' work *Voyager* incorporates a mode of interaction in which an improvising musician interacts with a computer that is generating music in response

[11] Anthony Braxton, "Keynote Address at the Guelph Jazz Festival, 2007." *Critical Studies in Improvisation,* Vol. 4, No. 1 (2008) : 2, DOI: 10.21083/csieci.v4i1.520

[12] Lee, Insuk, Eiru Kim, & Edward M. Marcotte. "Modes of Interaction between Individuals Dominate the Topologies of Read World Networks." *PloS one* 10, no. 3 (2015), DOI: 10.1371/journal.pone.0121248

to, and independent of, that improvisor's performance. George Lewis describes this work:

> Voyager [1,2] is a nonhierarchical, interactive musical environment that privileges improvisation. In Voyager, improvisors engage in dialogue with a computer-driven, interactive "virtual improvising orchestra." A computer program analyzes aspects of a human improvisor's performance in real time, using that analysis to guide an automatic composition (or, if you will, improvisation) program that generates both complex responses to the musician's playing and independent behavior that arises from its own internal processes."[13]

I considered *Voyager* as a model for the interactive exchange between creator, participant, and listener. Lewis is a creator of *Voyager* as he designed the interactive software. As true in many performance situations, the improvisor takes on the designation of all three roles. The improvisor is also a creator in contributing musical ideas through the exchange with the sound generated by the computer. The improvisor is a participant and listener as well because they are actively making sound in a realization of the work and actively listening to the sound emanating from the computer and their own performance. The mode of interaction between my own designation as a creator of *The Wunderkammer Interface*, and the creator/participant/listener role of the improvisor is similar. Lewis' work is nonhierarchical because of the equal exchange between the improvisor, the computer, and himself as creators. I consider *The Wunderkammer Interface* as nonhierarchical because of the equal exchange between the improvisor and I as creators through the digital processing of the improvisor's performance.

The difference in the designations of roles between *Voyager* and *The Wunderkammer Interface* is the role of the computer. In the instance of *Voyager*, the computer may be considered a creator, participant, and listener, while in *The Wunderkammer Interface* it is not. Lewis conceives a performance of voyager as

[13] George Lewis, "Too Many Notes: Computers, Complexity and Culture in “Voyager,” " *Leonardo Music Journal* Vol. 10 (2000): 33.

"multiple parallel streams of music generation, emanating from both the computers and the humans - a nonhierarchical, improvisational, subject-subject model of discourse, rather than a stimulus/response setup."[14] In *Voyager*, the activation of the computer program is not dependant upon a human participant, while *The Wunderkammer Interface* computer program is.

I am Sitting in a Room **by Alvin Lucier**

I Am Sitting in a Room by Alvin Lucier documents the process of recording a passage of spoken text, recording this passage played back into the same room, and recording each subsequent played back passage until the natural resonance of the room saturates the final renditions of the recorded material.[15] In this work, Lucier is the sole creator and participant. He is also the primary listener; Lucier invites other listeners to undergo the same a process of discovery he had undertaken when creating his work. *The Wunderkammer Interface* invites the participants to undergo a similar process of discovery. As the participant engages with the interface, a more saturated and distorted version of their own voice is presented back to them. Both *I am Sitting in a Room*, and *The Wunderkammer Interface* act as distorted sonic mirrors. In *I Am Sitting in a Room*, the listener enters an implied real physical space through the sonic reflection of Lucier; in *The Wunderkammer Interface*, the performer enters a imaginary space conjured from their own voice.

---

[14] George Lewis, "Too Many Notes: Computers, Complexity and Culture in "Voyager,"" *Leonardo Music Journal* Vol. 10 (2000): 34.

[15] Alvin Lucier, *Music 109, Notes on Experimental Music* (Middletown: Wesleyan University Press, 2012), 88.

***Rainforest* by David Tudor**

David Tudor's *Rainforest* consisted of four versions, realized between the years 1968 and 1973.[16] The common thread between each version is Tudor's activation, or instruction to activate, various resonant objects with transducers using generated or recorded samples of audio. Tudor's focus in creating music through the transformation of a sound, using the inherent resonant properties of an object acting as a filter, led me to consider creating a project in which different digital processing strategies could act as a compositional design in themselves, an inquiry that is a basic principle in the design of *The Wunderkammer Interface.*

In version IV, Tudor offers *Rainforest* as a collaborative effort in which other participants contribute various sounds and resonant objects to the project. *Rainforest IV* was presented for the first time at the 1973 New Music in New Hampshire three-week summer artist gathering as a participatory workshop. Tudor described the purpose of presenting his work in this manner as an act of giving it away after a period of working with it for a number of years.[17] The workshop participants were invited to use transducers he provided to activate objects found on their own accord using any sound with the caveat, "Rainforest IV, 1973, being coherent in its electronic principle, can accept any number of performers, and any kind of signal inputs (excluding only composed musics)."[18] *Rainforest IV* was initially presented, and has subsequently been

---

[16] Matthew Rogalsky, "Idea and Community: the Growth of David Tudor's *Rainforest*, 1965-2006." (Doctoral dissertation, City University London, 2006) 22-23.
[17] Matthew Rogalsky, "Idea and Community: the Growth of David Tudor's *Rainforest*, 1965-2006." (Doctoral dissertation, City University London, 2006) 190.
[18] David Tudor, quoted in Matthew Rogalsky, "Idea and Community: the Growth of David Tudor's *Rainforest*, 1965-2006." (Doctoral dissertation, City University London, 2006), 205.

presented, as a sound installation in which those observing it may move around the space and touch, or otherwise engage with, the activated objects.

Rainforest IV presents a mode of interaction that includes many creators. Tudor comments that "[Rainforest IV] is of an "open form", and the success of its performance depends upon consensus among performers as to what it means to participate in an "open form" piece."[19] *The Wunderkammer Interface*, also being an open form, presents the improvisor with a set of variable conditions through which he or she navigates. Success, in the case of *The Wunderkammer Interface*, occurs when the participating improvisor hears something meaningful in that navigation, and responds to it.

***Solo für Melodieinstrument mit Rückkopplung* by Karlheinz Stockhausen**

The composed aspects of *The Wunderkammer Interface* are the preconceived, fixed designs of digital audio processing that are triggered through the user interface. These preconceived designs, however, are not realized unless they are sonically activated through exterior sound generated by an improvisor who is making independent decisions regarding the creation of that sound. In being conscious of the processed sound, occurring in real time and being sonically present in the performing environment, the composed aspects of the Wunderkammer Interface inform the improvisor's creative impulse. The full realization of this work is not one or the other, but the combination of composed processing and performer's voice. A work that poses a similar type of relationship between preconceived processing and performer's voice is Stockhausen's work *Solo für Melodieinstrument mit Rückkopplung*. In this work, Stockhausen requires

[19] David Tudor, quoted in Matthew Rogalsky, "Idea and Community: the Growth of David Tudor's *Rainforest*, 1965-2006." (Doctoral dissertation, City University London, 2006), 204.

the solo instrumentalist to construct a performance scheme from provided score material. He specifies in the score, "In order to work out a version, one should best prepare 6 pages with empty systems…. According to the instructions of [step 5] one should then transfer into these systems in a chosen order the material from the note pages one decides to play… One may use any amount of material from the note pages (most of the time, a greater or lesser amount is left out)."[20] The material performed by the soloist is recorded and manipulated in real time by engineer assistants who follow a prescribed processing plan that manipulates the feedback produced by tape recorders playing back the recorded sound of the performer.

While *The Wunderkammer Interface* is a nonhierarchical mode of interaction, Stockhausen structures a hierarchy of roles. He exists as the creator of the work. The soloist is granted limited creative engagement, while acting as the main participant. The assistants are exclusively participants. The audience acts solely as listeners.

Modular Live Interface **by Sam Pluta**

Within the community of those who use and develop SuperCollider, and other digital processing environments, many use these tools for processing in live performance. Sam Pluta has developed his *Modular Live Interface* as: "a program designed for high level performance and infinite expandability. Because the software is written in SuperCollider, it can be programmed to do almost any task, but because the code is hidden behind a graphical user interface, lower level features do not burden the user in

[20] Stockhausen, Karlheinz. *Solo, Nr. 19: fur Melodieninstrument mit Rückkopplung*. (Vienna: Universal Edition, 1966), 14.

performance."[21] Pluta approaches his project as a versatile instrument for use in improvisatory musical contexts. While he is the primary architect of the *Modular Live Interface*, he grants the user of his work a great deal of choice in the operation of it in performance. Those who engage with *The Wunderkammer Interface* have that ability to make choices regarding its operation, but those choices are much more limited. The limitation that I place on the participant's ability to only trigger pre-programmed processing events lends to a definition closer to Lewis' nonhierarchical composition *Voyager* than the versatile instrument that Pluta offers in *The Modular Live Interface*.

The user of *The Modular Live Interface* may be one of many participants in a performance event, one in which the other participants may not engage with Pluta's interface. It is possible that *The Wunderkammer Interface* could be used in an improvisatory context in the same way that Pluta performs with *The Modular Live Interface*, a context in which a participant engages with *The Wunderkammer Interface* while other participants in the improvisatory context are not engaging with it.

#*[unassigned]* by James Saunders

The composer James Saunders provides, through his work, a model of using modular techniques as presented in industrial modular product design to develop musical form.[22] In his work *#[unassigned]*, Saunders presents the performers a collection of through composed, gestural action, and electronic modules to create a multidimensional

[21] Sam Pluta. "Laptop Improvisation in an Multi-Dimensional Space." (doctoral dissertation, Columbia University, 2012), 32.
[22] James Saunders, "Developing a Modular Approach to Music." (doctoral dissertation, University of Huddersfield, 2003), 56.

assemblage from which to construct a realization of his work. When reflecting where the identity of a composition lies in a modular work, Saunders offers:

> "The move towards working in an entirely modular way has many repercussions with regards to the compositional methodology and the identity of the resultant music. Conventionally composers produce pieces, which are discrete manifestations of their ideas at a given point in time. These ideas are continuous however (and separate from the pieces themselves) and often bleed across boundaries between pieces, but are necessarily constrained within individual works. With a modular approach, these boundaries still exist at the product (or version) level, but there is an additional segmentation of ideas and material at a modular level. The composition of a module is a bounded activity as there is a sense of completeness about it as an independent unit (a different situation to unbounded material in a non-modular piece)."[23]

Each module of *The Wunderkammer Interface* exists as a complete entity, while also referencing and allowing for the intersection with other modules. In both *#[unassigned],* and *The Wunderkammer Interface*, the participants in performance are given creative roles through the specific designs offered by the initial creators, Saunders and myself respectively.

Mode of interaction within *The Wunderkammer Interface*

Using the works presented here as models, I developed an interactive structure for *The Wunderkammer Interface* based on the fluidity of interaction and identity between the roles of creators, participants, and listeners. The outline of this structure is as follows: I am a creator of the music because I designed the interface that is used in this project's realization. The improvising musician is a creator as they too are forming musical ideas that are processed though the SuperCollider program in real time. The improvising musician is also the participant because they are making the sound that is being processed. The improvising musician is a listener as well who hears the sound coming

[23] James Saunders, "Developing a Modular Approach to Music." (doctoral dissertation, University of Huddersfield, 2003), 89.

from the speaker system and the sound they produce themselves. In a successful performance, the improvisor will respond to this sound, making a loop of creativity that gains momentum. Other listeners may include a person listening to a recording of the music or an audience attending the music event in which *The Wunderkammer Interface* is used. The result is a highly versatile work that expands the concept of an improvised performance.

Chapter 3: The Wunderkammer Interface

The Wunderkammer Interface is a nonhierarchical composition that invites an improvising musician to interact with twenty-five modules that activate pre-designed changes to different digital audio processing parameters. The improvisor may activate these modules at will, either as an isolated processing event or as overlapping events. In addition, the improvisor may trigger a series of modules that are automatically activated in a pre-programmed order. The pre-designed digital audio processing affects the sound from an input source chosen by the improvisor, and realizes the processed sound through a chosen output in real time. *The Wunderkammer Interface* was designed in the SuperCollider audio processing environment. The SuperCollider code that operates *The Wunderkammer Interface* is presented in Appendix 2.

The Interface

I designed *The Wunderkammer Interface* as a digital processing environment that is accessible for those both familiar and unfamiliar with digital audio processing. The architecture of *The Wunderkammer Interface* is presented to the improvisor as a set of twenty-five modules that appear in a single digital window (Fig. 2). A large, illustrated button represents each module. The window includes a brief description of the processing that takes place, and a guide to how that processing is depicted in the buttons. This description reads as follows:

> Thank you for using The Wunderkammer Interface
>
> This project is a live processing environment that is activated by pressing the white buttons above, or the blue buttons to the left. Each white button triggers a unique, minute-long, processing

scheme based on changes in the contours of a band-pass filter, and shifting amplitude modulations. The button guide above describes how this processing scheme is graphically represented in the buttons. Please note, that these representations are not meant to be an exact or specific graph of all the processing parameters that occur, but an identifiable depiction of the changing sonic characteristics for each button. The blue buttons to the left trigger multiple modules that are executed in sequence as a chain of events. Each series triggers a different type of progression inherent in the combined processing schemes as described.

To engage with the interface, the musician clicks on the different buttons in the window to activate the modules of processing. Limiting the improvisor's interaction to one action, - the pressing of buttons - is intended to speed up their transition from understanding how the interface works to experimenting with how the interface sounds.

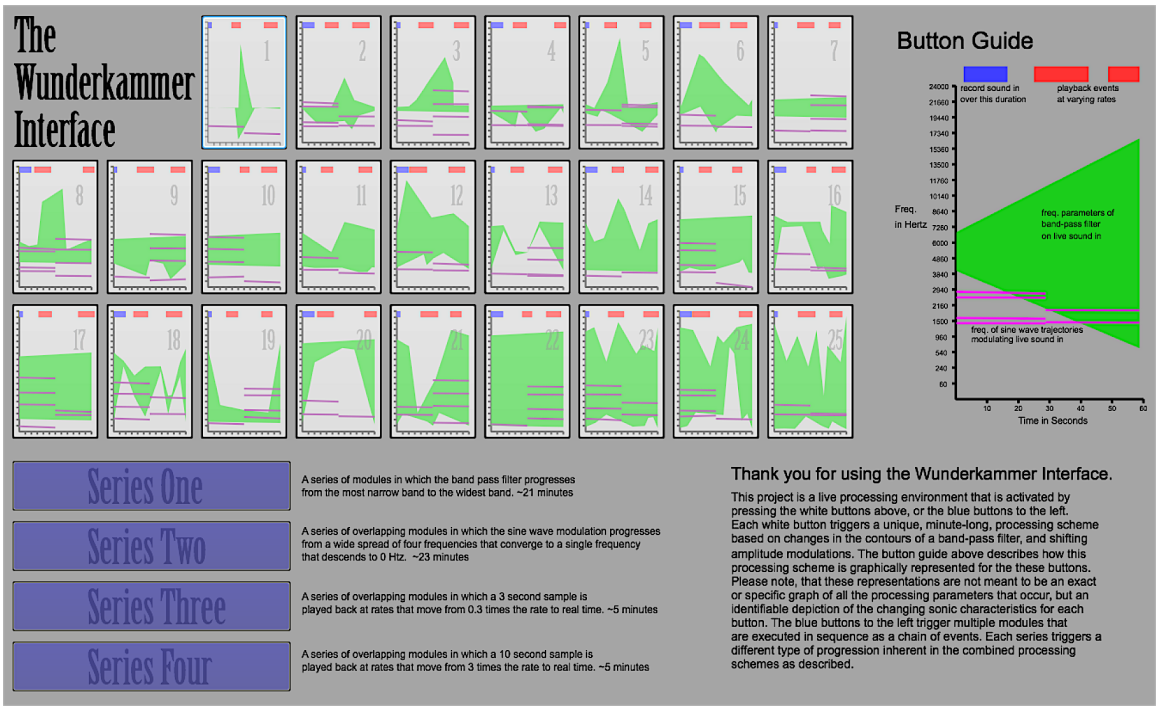

Figure 2 - *The Wunderkammer Interface* graphical user interface window

Three elements of digital processing are depicted in the buttons using a similar two-dimensional graph in which the x-axis represents time, and the y-axis represents frequency in Hertz (Fig. 3).

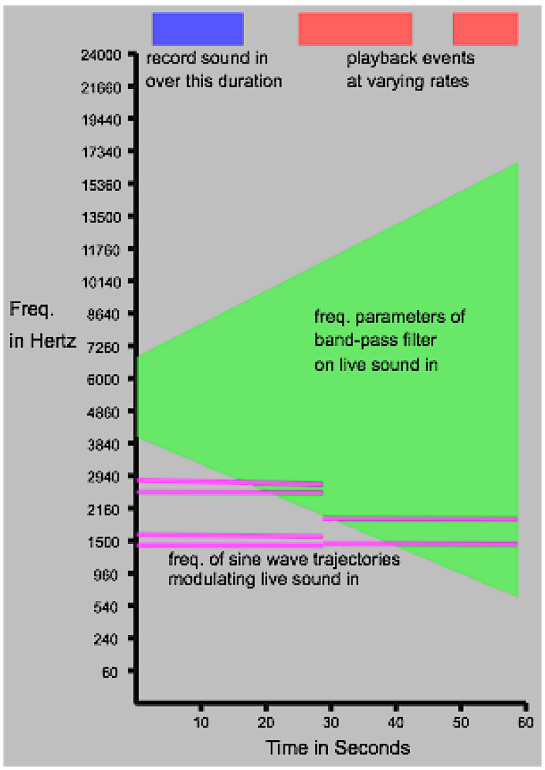

Figure 3 - Graph representing the processing elements common to every module

Each button activates a module of digital processing that is one minute long and incorporates processing elements that work within the same frequency range allowing for a similar graph for each module. The first element, a band-pass filter envelope that shifts in the range of its frequency band over the course of a module's one minute duration, is represented by a larger geometric shape that follows the contours of these changing frequencies. The second element is represented by straight lines that show the trajectory of sine wave glissandi that act as amplitude modulation on the filtered incoming sound. The third element is a sampling mechanism in which the initial seconds of incoming audio are recorded and played back as isolated events over the course of the module's

duration. A darker box at the top left of the button interface represents the recording duration. Lighter boxes to the right of the recording box represent the playback events. The uniformity of the graphic representations between each module allows the musician using the interface to quickly grasp the unique gesture profile for each module.

I intend for the improviser who engages with *The Wunderkammer Interface* to audibly experiment with the different components of the interface. While it is an accurate depiction of the digital processing that takes place, the visualization of these processing parameters leaves out indications of the exact values I specified within the SuperCollider code in order to present an interface of bold shapes that represent the audible contrast between modules that can be interpreted quickly. For example, the lines that represent the sine wave amplitude modulation only outline the straight trajectory of the modulation from the onset frequency to an endpoint frequency. The contours of the envelope as specified in the SuperCollider code, are not displayed. The difference between the trajectory of the modulation and the modulation envelopes can be seen in Figure 7. The additional processing elements of shifts in reverberation, shifts in amplitude, and differences of pan position are not depicted at all in order to preserve the simplified clarity presented in the interface. It was also my intent to incorporate dimensions of processing not visible to the improviser for the improvisor to aurally discover in performance.

The rectangular buttons in the lower left of the interface window trigger pre-ordered series of modules that are organized to reveal larger progressions inherent in the design of the digital processing (Fig. 4). The button labeled Series One, for example, plays the modules in the order they are presented over the course of twenty-two minutes.

In this order, the frequency range for the band-pass filter envelopes increase from their narrowest range at the onset of module 1 to their widest range at the conclusion of module 25. These series triggers also allow an improvisor to activate longer durations of processing without needing to activate a new module after one minute of performance.

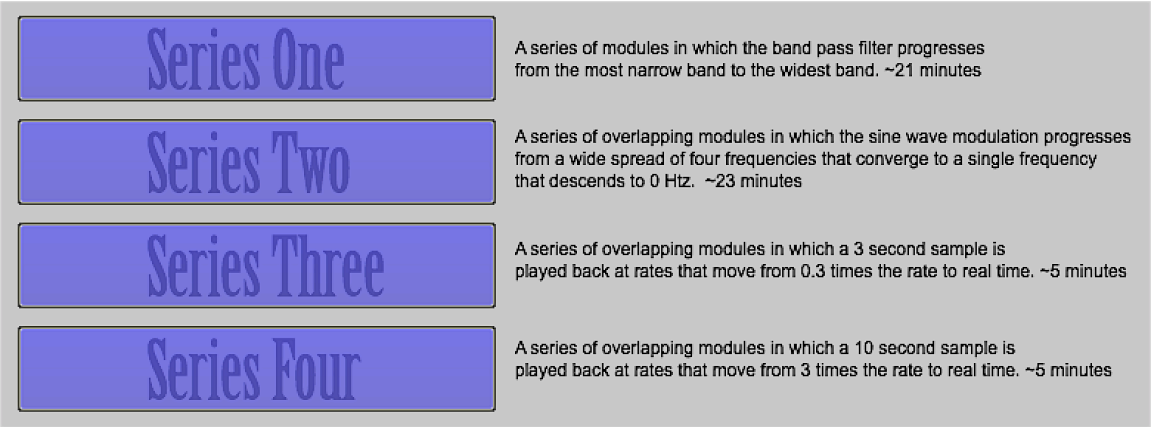

Figure 4 - Buttons that trigger pre-programmed order series

Processing Scheme Shared between Individual Modules

Each module of *The Wunderkammer Interface* changes the parameters of the incoming audio signal with the same digital processing scheme (Fig. 5). The duration of each module is uniform, each processing the audio signal over the course of one minute. This audio processing scheme can be divided into two processes that occur in parallel over the course of one minute. In the first, the signal is constantly audible as different processing parameters affect the sound. In the second process, a sample of audio is recorded in the first seconds after a module is activated; this sample is played back twice over the course of the one-minute module duration. Changes in the rate of playback and

the application of reverberation sonically alter these samples from the original recorded sound.

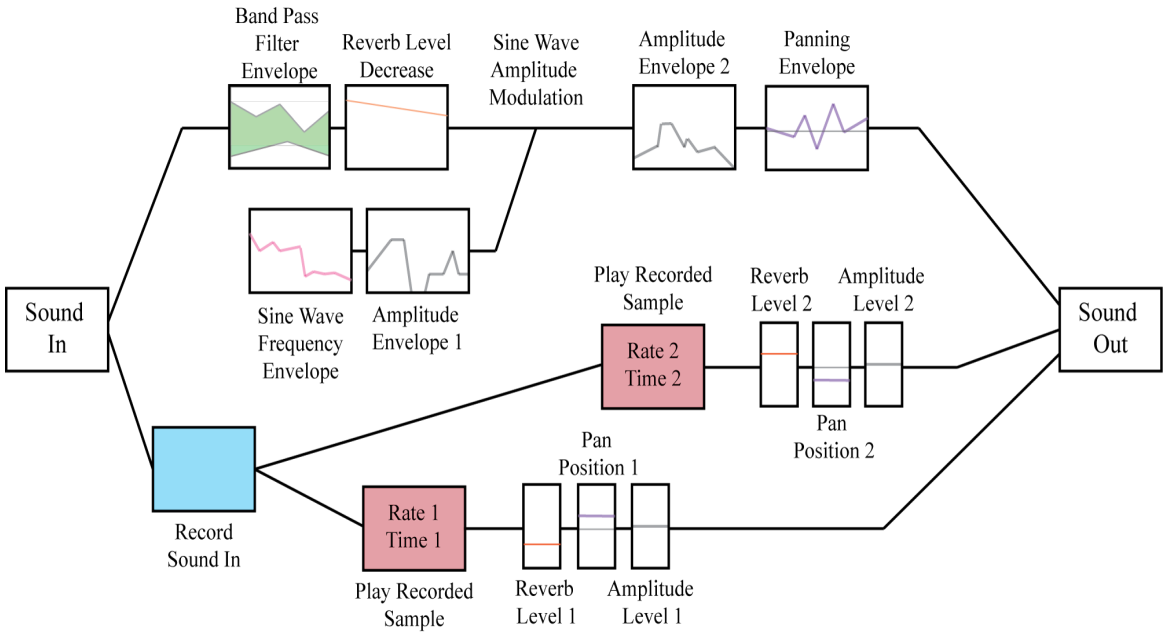

Figure 5 - Processing scheme shared between individual modules

A detailed description of the first process is as follows. The spectrum of the incoming audio signal is altered through the shifting frequency parameters of a band-pass filter envelope. While the fluctuation of these parameters change from wide to narrow, the initial envelope range that affects the audio signal at the beginning of the one-minute duration is narrower than the range at the end of that minute. The effect of this technique is a subtle audible emergence in the processed sound for each module. In order to enhance this audible emergence, a gradual decrease in the level of reverberation of that filtered signal occurs over the course of the module's duration. The amplitude of this filtered, reverberating signal is then modulated with a single sine wave or set of sine

waves. The frequencies of these sine waves change over the course of the one-minute module duration in two distinct stages of frequency shifts. In the first thirty seconds, the frequencies deviate slightly from a gradual trajectory, then shift and deviate in the last thirty seconds from a new trajectory. This shift is audible in the processed sound as a dramatic swooping gesture at the thirty-second mark of each module. A separate amplitude envelope dictates the amount that this sine wave acts upon the initial filtered, reverberating audio. In the final steps of this first process, a second amplitude envelope shifts the volume of this modulated signal and a panning envelope changes the location of this signal in a stereo speaker field.

The second process records a sample of the incoming audio in the initial seconds after a button is triggered and plays back that audio at varying rates. In the even-numbered modules of *The Wunderkammer Interface*, the first ten seconds of incoming audio are recorded, and in the odd-numbered modules, the first three seconds are recorded. For the modules that record a ten second audio sample, the rates of each playback event are set at different higher values than the original sample resulting in sound events that are higher in pitch and shorter in duration. For the modules that record a three-second audio sample, the rates of each playback event are set at different lower values that the original sample resulting is longer, lower sound events. In all modules, this recorded audio is played back twice over the course of the module's one-minute duration. The points at which these playback events occur over the course of the module's one-minute duration are not uniform between modules in order to obscure the establishment of a regular occurring loop. As a rule, however, the first event will occur

within the first thirty seconds, and the second within the last thirty seconds. A different level of reverberation and a different pan position are set for each playback event.

While the processing elements are the same for each module of *The Wunderkammer Interface,* the variation between modules exists in the different values that dictate the shapes and trajectories of the processing envelopes. By limiting myself to changing these values within the processing scheme template outlined previously, I achieved a satisfying level of contrast between modules while maintaining a sense of cohesion as a collection. Furthermore, working within a similar template between modules allowed me to make comparisons between them, and execute effective changes that resulted in a more nuanced variety. The following are some examples of the compositional strategies I employed to determine different envelope values for each module.

The contrasting shapes of the band-pass filter envelopes in each module are deviations from a larger wedge that begins at the onset of module 1 and ends at the conclusion of module 25. These deviations are dictated by my distribution of different node schemes for the high-pass and low-pass thresholds of the band-pass filters. In module 9 for example, the low-pass threshold envelope incorporates no nodes of deviation, while the high-pass threshold envelope incorporates four nodes of deviation from the initial trajectory (Fig. 6). In module 23, the low-pass and high-pass envelopes both incorporate six nodes of deviation (Fig. 6). The distribution of node schemes is designed to create distinguishable variety in the module collection.

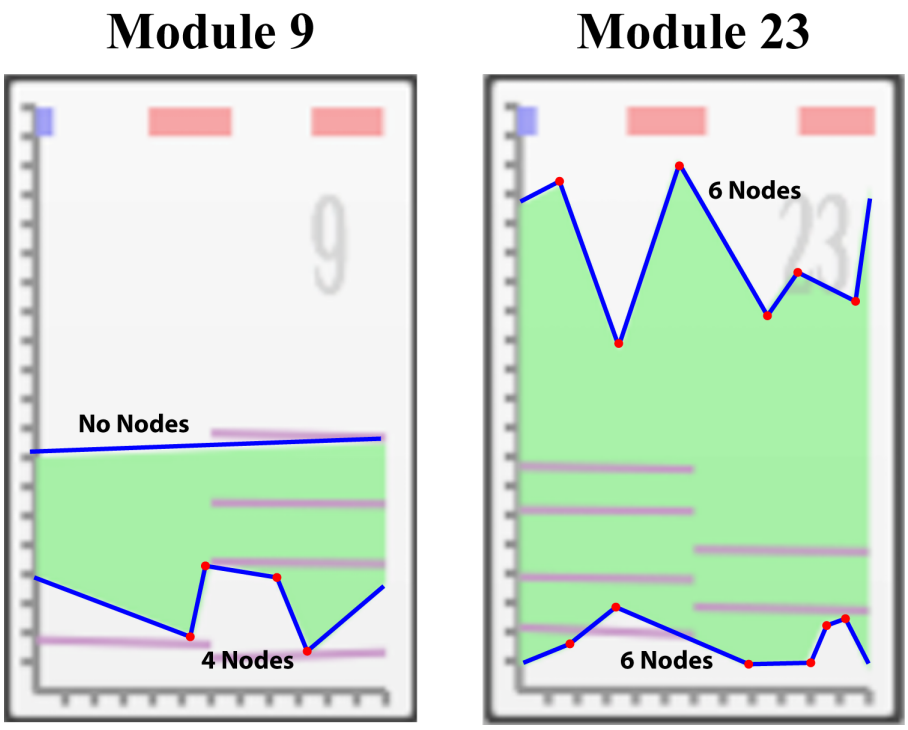

Figure 6 - Nodes of deviation in band-pass filter envelopes

In each module there is a shift in the trajectory of the sine wave modulation envelopes that occurs at thirty seconds. I purposefully calibrated this shift to occur at the same point for each module in order to incorporate an element of uniformity amongst the contrasting parameters of deviation between each module. The distribution of amplitude modulation trajectories is unique for each module. For example, the audio signal is modulated by a set of four frequency trajectories in the first thirty seconds of module 10 (Fig. 7). These frequency trajectories are: 4197 to 4032 Hz, 2406 to 2379 Hz, 1062 to 1020 Hz, and 159 to 187 Hz. A modulating frequency trajectory set, like what is heard in the first thirty seconds of module 10, is part of a fragment of an convergence of frequencies, implying a larger wedge shape in the overall form of the modules as a collection. In the last thirty seconds of module 10, there is only one trajectory of the

modeling frequency that moves from 60 Hz to 30 Hz (Fig. 7). I vary the contrast in these trajectory shifts from module to module. In Module 14 for example, this shift is less dramatic from a single frequency trajectory that moves from 211 Hz to 181 Hz to a single trajectory that moves from 362 Hz to 331 Hz (Fig. 7).

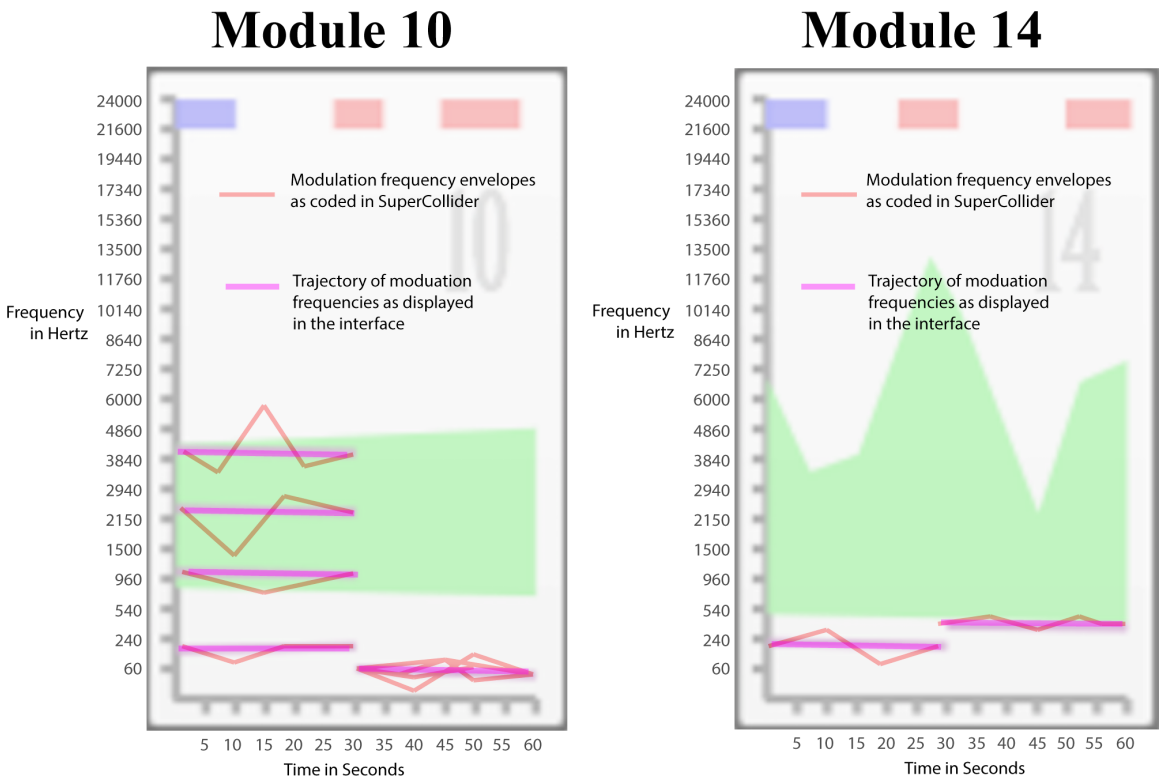

Figure 7 - Amplitude modulation frequency envelopes

Progression

Designing *The Wunderkammer Interface* as a set of modules allowed me to develop multiple avenues of progression. These avenues of progression may be heard by combining modules in different orders. I designed four potential avenues of progression: the gradual increase in the range of the band-pass filter, the gradual convergence of the

modulating frequency trajectories, and two convergence trajectories that dictate the rate at which the recorded audio samples are played. The buttons in the lower left of the interface trigger these different avenues of progression.

The larger wedge progression that occurs in the band-pass filter envelopes can be seen in the order of the modules as they are presented in the interface. At the onset of module 1, the range of the band-pass filter is at its narrowest, between 1980 Hz and 2000 Hz; at the conclusion of module 25, the range is at its widest, between 10 Hz and 22000 Hz (Fig. 8). While nodes change the direction of the band-pass frequencies within each module, the last value of the low-pass frequency limit will always be higher than the first value, and the last value of the high-pass frequency limit will always be lower than the first value in every module.

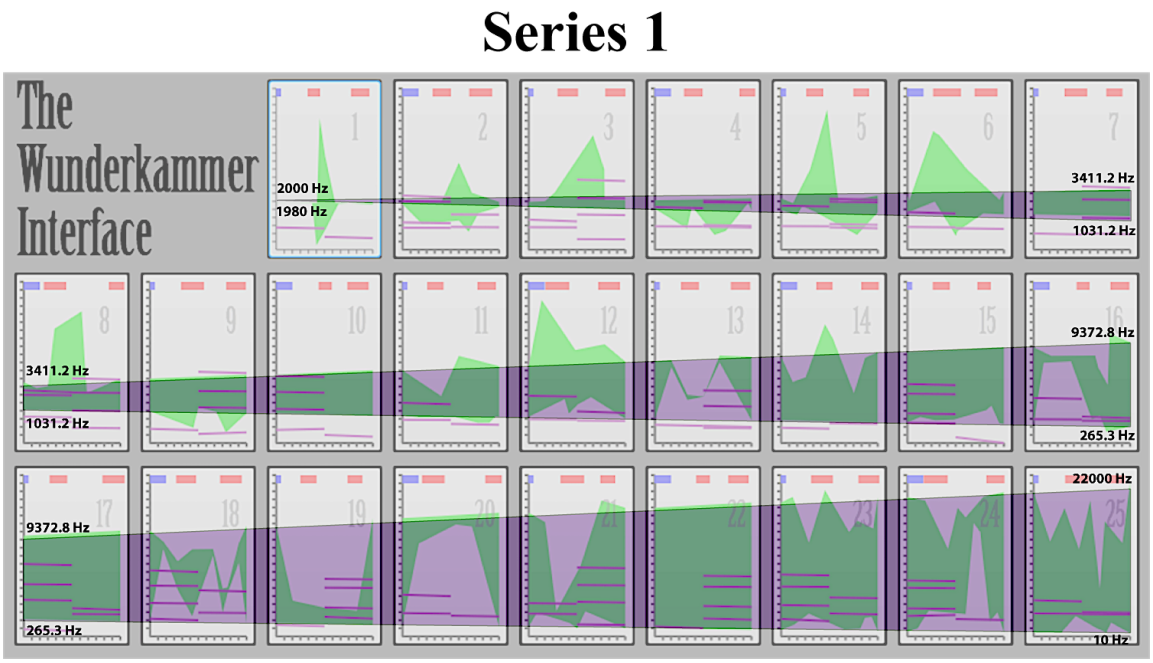

Figure 8 - Band-pass filter expansion over the course of the 25 modules

Over the course of the twenty-five modules, the reverberation of the incoming audio signal is calibrated to gradually decrease. At the onset of module 1, the dry/wet balance, room size, and high-frequency damping are set to their highest levels, while at the conclusion of module 25, they are set at the lowest levels. An increase in the range of the audible frequency spectrum, in combination with a decrease in reverberation of that audio signal, is the basic processing scheme that implies a sound moving from a distance to a proximity closer to the listener.

A convergence of frequency dictates the trajectories heard in the amplitude modulation. When the modules of *The Wunderkammer Interface* are re-ordered, and aligned so that they overlap to create a point of intersection at the thirty-second mark, this progression can be heard (Fig. 9).

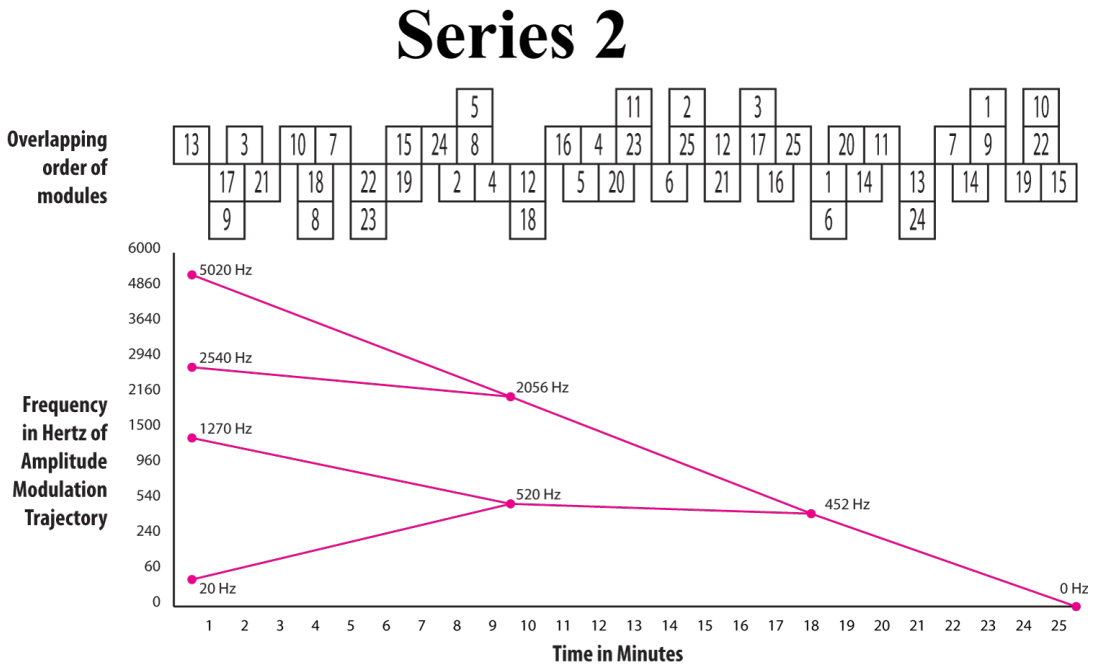

Figure 9 - Larger convergence of amplitude modulation frequencies

At the thirty-second mark of module 13, the set of four modulating frequencies are calibrated to their largest spread, those frequencies being 5020 Hz, 2540 Hz, 1270 Hz, and 20 Hz at that thirty-second mark. Over the course of the overlapping order of modules, the thirty-second long sections of modulating frequency trajectories gradually converge and descend to 0 Hz, reached at the conclusion of module 15. Pressing the Series 2 button in the interface activates this progression. I distort a clear realization of this progression in the following ways: I maintain the presence of overlapping trajectories from other modules that distort the larger convergence trajectory, I deviate from these frequency trajectories within the scope of individual modules, and I offset the activation of these modules from the regular 30 second intervals as presented in Figure 9 in order to convolute any sense of regularity in this progression.

Two avenues of progression exist in the trajectories that dictate the rate at which the recorded samples are played back in each module. As specified previously, a sample of the incoming audio is recorded at the beginning of each module. This sample is played back twice over the course of the module. Each playback event is set to occur at a different rate. The modules can be re-arranged so that these rates form a trajectory that leads from a playback rate ratio that has a high deviation value from the original recorded rate to a playback event that occurs at the original recorded rate. This trajectory can occur from two directions, one in which the playback rates are at lower values that the original rate and one in which the playback rates are at higher values.

Figure 10 shows how the overlapping orders of modules convey these progressions as triggered by the Series 3 and Series 4 buttons. In Series 3, the rate of the first playback event in module 25 is set to 0.313 times the original rate. The rate of the

last playback event in module 15 is set to 0.875 times the original rate. By following the rates in the smaples that occur between modules 25 and 15, a trajectory towards a playback rate equal to the original rate can be seen. In Series 4, the rate of the first playback event in module 16 is set to 3.182 times the original rate. The rate of the last playback event in module 12 is set to 1.014 times the original rate. Following the rates that occur between modules 16 and 12 reveals a parabolic trajectory towards a playback rate of 1.

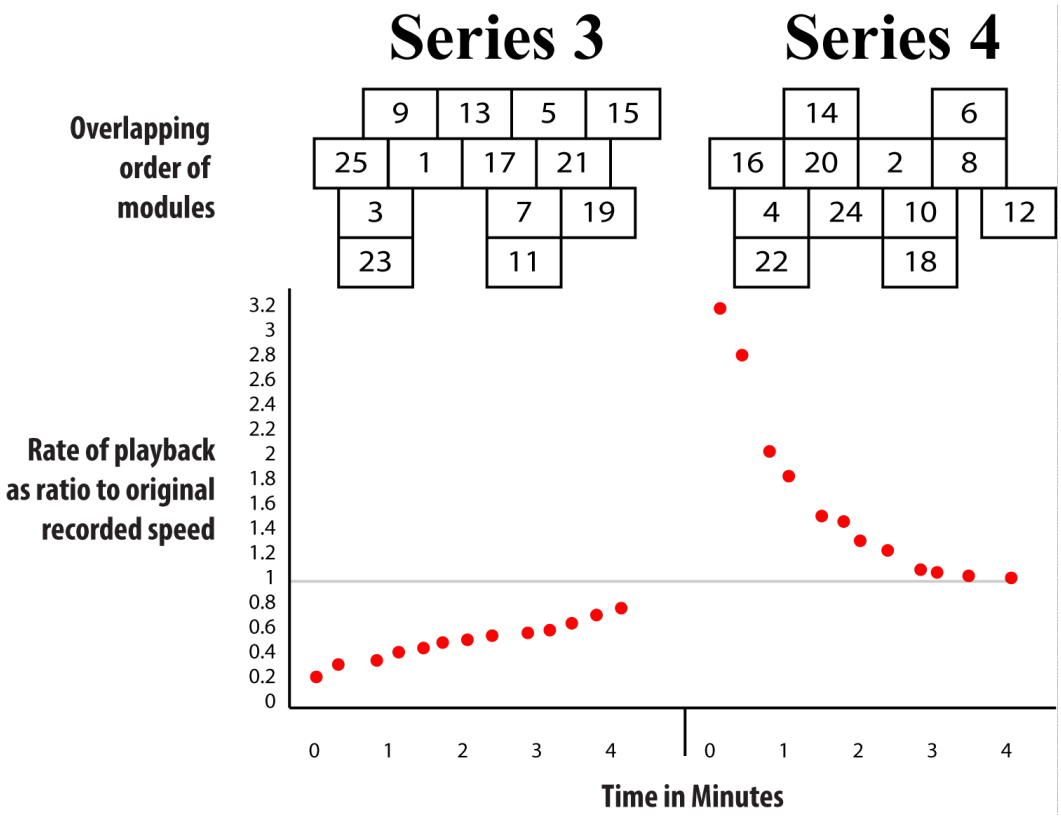

Figure 10 - Rate trajectories in Series 3 and Series 4

The playback events in every module of *The Wunderkammer Interface* can fit within one of these two trajectories. I chose, however, to only use a selection of modules to create the series triggered by the Series 3 and Series 4. This gives the improviser the

option to activate series that lasts for approximately five minutes, significantly shorter than the duration of the series activated by the Series 1 and 2 buttons.

These avenues of progression do not need to be realized in their entirety to give those listening to *The Wunderkammer Interface* a compelling sense of these larger progressions. The larger wedge-like trajectories of the band-pass filter range and the frequencies of the amplitude modulation create this compulsion even within individual modules. Participants who interact with *The Wunderkammer Interface* do so by activating modules at their discretion in an improvisatory manner, not necessarily in the orders outline here. The varied contours within individual modules and fragmented realizations of the larger progressions are meant to give the sense of progression that loops upon itself in perpetuity.

Chapter 4: Performance Profiles

I asked the following improvising artists to make recordings with *The Wunderkammer Interface*. The different approaches they took demonstrate the contrasting variety that can result in its performance. While the processing from *The Wunderkammer Interface* transforms their voices in these recordings, each performer dictates the manner in which music was made using the techniques integral to their personal style of improvisation. It is, however, audible in these performances that the digital processing leaves a signature of design that can be identified as *The Wunderkammer Interface*. These performances can be heard in the recordings that accompany this dissertation. A guide to these recordings can be found in Appendix 1.

Wind Chimes reference tracks

To demonstrate the audible design of *The Wunderkammer Interface* processing, I made recordings of the different modules and series processing the sound of a set of wind chimes that were continuously ringing. The wind chimes acted as a static sound source so that contrast and progression that is inherent in these digital processing schemes were clearly audible. Track 1 demonstrates what each module sounds like in isolation, while tracks 2, 3, 4, and 5 demonstrate how each of the four series operate. In track 6, I am activating buttons at will to demonstrate a more improvisatory interaction with *The Wunderkammer Interface*. In each of these recordings, the listener can clearly follow the contour of the gestures that result from the various envelopes of processing, and can hear

how the larger progressions are realized without the improvisatory input from a performer.

Yan Pang

The recordings that pianist Yan Pang made with *The Wunderkammer Interface* demonstrate how the sound from a fixed-pitch instrument, like the piano, transforms when the shifting frequency envelopes of the digital processing are applied to it. The overlapping bands of amplitude modulation scramble the spectral signature of the piano's acoustic resonance. The short articulations in Pang's playing resemble the more percussive sound of virtual metallophones. The melodies she plays become rhythms, like those in John Cage's music for prepared piano. Throughout the different tracks presented here, Pang takes advantage of sustaining the piano's acoustic resonance over longer periods of time allowing for the modulation envelopes to shape the natural decay of that resonance. The dramatic shifts that occur in this modulation, at the thirty-second mark in each module, are prominent in the sustained gestures that Pang executes.

Pang also takes advantage of incorporating different harmonies within the processing context. Playing, for example, a 7th chord over a collection of multiple modulation envelopes that move in different directions. As a result of this combination, the intervallic relationships identified in the 7th-chords shift in and out of focus similar to the visual blurring that occurs while looking at an object at the bottom of a pool where the surface of the water is slightly disturbed. The interplay between the sound of Pang's playing in real time and the sounds of the recorded samples create an additional audible illusion. The different levels of reverberation that affect Pang's playing in real time

versus the levels of reverberation that affect the recorded samples juxtapose these sound events at different positions of foreground and background in the listener's sense of auditory space. Because the playback events sound much closer in the listener's perception than Pang's real time sound at times, the sense of what Pang is actually playing, and what she is not, becomes distorted.

Jay Afrisando

Jay Afrisando recorded two extended improvisations with *The Wunderkammer Interface* playing a Korean piri with a saxophone mouthpiece and a saluang, an end-blown flute native to West Sumatra. In contrast to Pang's performance, there is a distinction between the acoustic sound of Afrisando's performance and the processed sound coming from the computer. At the beginning of each improvisation, Afrisando incorporates the processing gradually, activating modules in isolation. The fading in of these initial isolated modules is clearly audible as a separate event from the acoustic piri and saluang. As Afrisando progresses farther into each improvisation, he activates an increasingly saturated texture of processed sound. This sound blends with what he is playing acoustically, creating the impression of instruments slowly transforming over the course of the improvisation. Later, in both improvisations, Afrisando activates multiple modules in rapid succession. As a result the overlapping recorded sample events create sudden bursts of sonic activity.

The higher register and simpler sonic spectrum of the piri interacts with the amplitude modulation envelopes in a unique way. A majority of the amplitude modulation frequencies are lower than the fundamental frequency of the piri's sonic

spectrum. Since the shifting modulation frequencies are lower, they become clearly defined and distinct because they act as a new fundamental of the combined sonic spectrum. Afrisando explores the relationship between the acoustic sonic spectrum of his instruments and the affect that the processing has on them in his improvisations. This effect, for example, is even more pronounced when Afrisando purposefully overblows the saluang, activating the instrument's harmonics that are at the highest extent of the audible overtones. Afrisando also makes reference to the audible beating that occurs in the close intervallic relationships that emerge from the layering of recorded samples with similar rates through the heavy vibrato that he is capable of producing on both instruments.

Benjamin Cold

Benjamin Cold uses short, melodic fragments in his improvisations. He uses the processing schemes of *The Wunderkammer Interface* to connect these shorter melodic ideas. When these fragments are recorded and played back at different rates, the samples sound similar to the original acoustic saxophone, only transposed. The higher and lower manifestations of these fragments link together to form longer melodic lines with distinctive contour. At different points in his improvisations, Cold uses this technique in a more condense manner by overlapping multiple modules. This generates a fugue-like polyphonic texture. Between melodic fragments, Cold incorporates longer sustained notes. Because Cold has the capability to maintain a more robust tone and higher dynamic level over the course of these sustained gestures, the envelopes of amplitude modulation are audible almost in their entirety when acting on these sustained notes. The

amplitude envelopes become a second melodic contour that works in parallel with the chains of melodic fragments occurring simultaneously.

The way in which the overlapping modules realize the larger progressions inherent in the design of *The Wunderkammer Interface,* either in their entirety or in fragments, is audible in Cold's performance. When the fragments of melody are recorded and played back, for example, the trajectory of the different rates at which these playback events occur is reflected in the arching contours that emerge when these fragments are linked together. The larger convergence of amplitude modulation can be heard in Track 14, where the higher, more spread out, frequencies of the modulation are prominent at the beginning of the improvisation, while more condensed clusters of frequencies are prominent at the midpoint of this track.

Ron Coulter

Ron Coulter recorded four tracks with *The Wunderkammer Interface*; each incorporates a different set of percussion instruments. The different instruments he uses for these tracks are: elephant bells for track 20, bird calls and an ocarina for track 21, a tam-tam in track 22, and a drum set in track 23. The digital processing from *The Wunderkammer Interface* affects each instrument set up differently. The timbre of the snare drum in track 23, for example, does not reveal the shape of the modulation envelopes when they are applied to it because of the lack of overtones in that acoustic sound. Instead, the processing of the snare creates a sizzling effect. Coulter uses this effect by creating waves of this fuzzy distortion by increasing his activity on the snare drum, then expanding to other parts of the drum set at the crest of the wave. In contrast,

the tam-tam has a timbre very rich in overtones; As a result, the shapes of the envelopes are clearly present. Because many of the instruments that Coulter uses have a more noise-based spectral signature, the effect of the band-pass filter envelopes is more prominent that with the pitched instruments used by other improvisors. For example, when the band-pass filter envelopes are applied to the sound of the bird call in track 20, the shifting band-pass envelopes can be heard in the appearance and disappearance of the higher spectrum of that acoustic sound.

Coulter uses the events that occur in the processing as cues to create new gestures on the instrument he has at hand. Simultaneously, Coulter's strong sense of pacing dictates the progression of the improvisations. In track 20, for example, he sets up textures of sampled sound by playing rapid gestures on the elephant bells that are picked up in the recording mechanisms that are part of each processing module; he allows time for these recorded samples to play later in the duration of the performance. Coulter then creates a new gesture, using a different technique of playing the bells, while the samples of his previous gesture is being played back. Coulters approach demonstrates how the improvisor can shape not only the sound realized in *The Wunderkammer Interface*, but also the progression of a performance with it.

Marcelo Rilla

In track 24, Marcelo Rilla processes the sound of his speech. He improvises a Spanish translation of the instructions of *The Wunderkammer Interface* window and makes commentary regarding his process of learning how the interface operates. His use of speech gives the illusion of a theatrical space with many characters. In this

improvisation, Rilla activates many modules in short succession, creating layers of processing that deeply distort the sound of his voice as the listener hears it in real time. The recorded samples become the most decipherable part of Rilla's improvisation, but because these samples occur at different rates, these voices sound like they come from different people. The orientation of these samples at different places in the stereo field increases this effect. As the improvisation progresses, the different voices emanating from Rilla's performance with *The Wunderkammer Interface* gives the illusion of an increasing number of people joining an intensifying argument.

In track 25, Rilla again uses the act of learning how to operate *The Wunderkammer Interface* as his material, but now records the machine noise created by his computer as he clicks on the interface buttons. The overarching form of this improvisation resembles the overarching wedge shapes inherent in the interface's digital processing design. As the improvisation builds, envelope contours and sampled material become increasingly recognizable.

Samu Gryllus

Samu Gryllus created a set of recordings in which he used *The Wunderkammer Interface* to transform sounds from his life that occur outside of his professional music practice. In track 26, he uses a ceramic mug and a toy keyboard as well as simple vocal gestures, speech, and simple body percussion like clapping and snapping to create a collage of sound. In Track 27, Gryllus uses *The Wunderkammer Interface* to process the environmental sounds occurring around him, creating an enhanced soundscape. Gryllus recorded the final track with his son whose vocalizations and claps are processed with the

interface. In addition to its use in more focused improvisatory performance, *The Wunderkammer Interface* is well suited for more informal use as demonstrated by Gryllus. In these recordings, Gryllus enjoys the capability to take on the roles of creator, participant, and listener as is possible through *The Wunderkammer Interface.*

Chapter 5: Conclusion

The early Wunderkammern were some of the initial vehicles through which the scholarly elite of the sixteenth and seventeenth centuries re-envisioned their ability to master their own universe using their knowledge of it. Instead of only being subject to a greater divine Creator, the artifacts and intricate automatons that appeared in these collections reflected a recognition that humans too have had, and can take on, the identity of creator. Amassing these great collections of both natural and human-made objects represented a renewed participation in the attainment of knowledge, and the invitation to view these collections offered the observer the opportunity to share in that knowledge. These Wunderkammern were models of a new mode of interaction that sixteenth and seventeenth century scholars used to interact with their universe.

Guerino Mazzola describes this shift as the reality of an active, dynamic body of knowledge that develops through interaction:

> "The change in this dynamic system does not occur through the autonomous action of a 'world automaton'. It is the result of an incessant and substantial interaction of mankind with the corpus of knowledge. The world of knowledge in the "viewer model" also suffers "shipwrecking"[24] because we constantly intervene in it actively. Like the political and civilizing dynamics, or the simple continental shift, the encyclopedic corpus is an open system that is uninterrupted and reinvented in synergy with human knowledge production. The metaphor of the speculum, that is, a passive vision of given things, does not apply. We do not consider prefabricated, pre-established things and concepts, but we continually re-define them, add new knowledge to old and correct defects. It is thus the passive metaphor of the speculum to replace by a more adequate interactive metaphor of an instrumentum. The latter corresponds to the computer as an instrument of an interface, which is active in both directions: from databases of knowledge to humans and vice versa. This aspect is substantial, since it also questions the real nature of knowledge. Concepts in this context are no longer any worldly beings (ideas) that have been established once and for all in a Platonic heaven, uperouranioz topoz (hyperouranios topos). Rather, they represent operational units whose ontology is at the very least linked to the accessibility to conceptual

[24] Hans Blumenberg. *Shipwreck with Spectator*. trans. Steven Rendall (Boston: MIT Press, 1996).

> determinations. "Understanding of a concept in this perspective has much to do with what is activated when one is faced with the task of understanding what one means to understand.[25]

The Wunderkammer Interface is a work that models Mazzola's representation of knowledge. A performance of *The Wunderkammer Interface* is an incessant interaction between computer and performer. It is an open system that is reinvented with each performance. It offers a structure with which to interact, yet allows decisions to be imprinted upon it.

The performances described in this paper demonstrate unique realizations of *The Wunderkammer Interface* through different interactions with it. As a growing collection, the performances created with *The Wunderkammer Interface* exist as a Wunderkammer of sound. The improvisors engaging in the creative process through *The Wunderkammer Interface* take part in a process of discovery similar to the quests of knowledge attainment undertaken by the viewers and collectors of previous Wunderkammern. *The Wunderkammer Interface* invites those who engage with it to discover new music through their own voice, a process of improvisation that evolves continuously.

[25] Guerino Mazzola, "EncycloSpace - the Knowledge space in the information age," trans. Benjamin Klein, *Humanities @ Encyclospace,* access date April 6, 2017, http://www.encyclospace.org/special/encyclospace-CD/html/chapter11/main.html

Bibliography

Blumenberg, Hans. *Shipwreck with Spectator*. trans. Steven Rendall. Boston: MIT Press, 1996.

Braxton, Anthony. "Keynote Address at the Guelph Jazz Festival, 2007." *Critical Studies in Improvisation* 4, No. 1 (2008) : 1-13, DOI: 10.21083/csieci.v4i1.520

Bredekamp, Horst. *The Lure of Antiquity and the Cult of the Machine: The Kunstkammer and the Evolution of Nature, Art, and Technology*, trans. Allison Brown. Princeton: Marcus Wiener Publishers, 1995

Bruckner, Max. *Vielecke und Vielflache: Theorie und Geschichte*, Leipzig: B. G. Treubier, 1900.

Campbell, Brenna, Élodie Lévèque & Erin Jue. "Marcel Duchamp's Boîtes-en-valise: Collaboration and conservation." *Studies in Conservation,* 57 (2012): sup1, S52-S60, DOI: 10.1179/2047058412Y.0000000017

Dillon, Brian. "Ugly Feelings," in *Damien Hirst*, ed. Ann Gallagher, 21-29. London: Tate Publishing, 2012.

Gallagher, Ann. "Introduction," in *Damien Hirst*, ed. Ann Gallagher, 11-19. London: Tate Publishing, 2012.

Gregory, Richard. "Trumping Eyes, Part 2" *Perception*, 38 (2009): 1265-1266, DOI: 10.1068/p3809ed

Klein, Benjamin J Mansavage. "Speech, rotation, tuba: a Progression of Works, 2008-2010." Master's thesis, Wesleyan University, 2010.

Lee, Insuk, Eiru Kim, & Edward M. Marcotte. "Modes of Interaction between Individuals Dominate the Topologies of Read World Networks." *PloS one* 10, no. 3 (2015), DOI: 10.1371/journal.pone.0121248

Lewis, George. "Too Many Notes: Computers, Complexity and Culture in "Voyager,"" *Leonardo Music Journal* 10 (2000): 33-39.

Lucier, Alvin. *Music 109, Notes on Experimental Music*. Middletown: Wesleyan University Press, 2012.

Mazzola, Guerino. "EncycloSpace - the Knowledge space in the information age," trans. Benjamin Klein, *Humanities @ Encyclospace,* access date April 6, 2017, http://www.encyclospace.org/special/encyclospace-CD/html/chapter11/main.html

Muensterberger, Werner. *Collecting, an Unruly Passion*. Princeton: Princeton University Press, 1994.

Pluta, Sam. "Laptop Improvisation in an Multi-Dimensional Space." Doctoral dissertation, Columbia University, 2012.

Robinson, Joyce Henri. "An American Cabinet of Curiosities," in *Acts of Possession, Collecting in America*, ed. Leah Dilworth, 16-41. New Brunswick: Rutgers University Press, 2003.

Rogalsky, Matthew. "Idea and Community: the Growth of David Tudor's *Rainforest*, 1965-2006." Doctoral dissertation, City University London, 2006.

Saunders, James. "Developing a Modular Approach to Music." Doctoral dissertation, University of Huddersfield, 2003.

Stockhausen, Karlheinz. *Solo, Nr. 19: fur Melodieninstrument mit Rückkopplung*. Vienna: Universal Edition, 1966.

Appendix 1: Recording Guide

Recordings of *The Wunderkammer Interface* can be found at the following locations:
ProQuest Supplementary Material

Reference tracks
recorded February 12, 2017 at the University of Minnesota - Minneapolis, Minnesota

Track 1 - demonstration of individual modules, wind chimes
Track 2 - demonstration of Series 1, wind chimes
Track 3 - demonstration of Series 2, wind chimes
Track 4 - demonstration of Series 3, wind chimes
Track 5 - demonstration of Series 4, wind chimes
Track 6 - improvised module activation, wind chimes

Yan Pang
recorded April 10, 2017 at the University of Minnesota - Minneapolis, Minnesota

Track 7 - piano
Track 8 - piano
Track 9 - piano
Track 10 - piano
Track 11 - piano

Jay Afrisando
recorded April 10, 2017 at the University of Minnesota - Minneapolis, Minnesota

Track 12 - piri with a saxophone mouthpiece and saluang
Track 13 - piri with a saxophone mouthpiece and saluang

Benjamin Cold
recorded March 20, 2017 at the University of Minnesota - Minneapolis, Minnesota

Track 14 - alto saxophone
Track 15 - alto saxophone
Track 16 - alto saxophone
Track 17 - alto saxophone
Track 18 - soprano saxophone
Track 19 - soprano saxophone

Ron Coulter
recorded March 21, 2017 at Caspar College - Caspar, Wyoming

Track 20 - elephant bells
Track 21 - audubon bird call, turkey call, clay ocarina
Track 22 - tam-tam, various implements
Track 23 - drum set

Marcelo Rilla
recorded March 30, 2017 in Montevideo, Uruguay

Track 24 - spoken text
Track 25 - electronics

Samu Gryllus
recorded February 18 and 19, 2017 in Hamburg, Germany

Track 26 - misc. household objects, vocalization
Track 27 - environmental sound
Track 28 - clapping, snapping, vocalization

Appendix 2: Supercollider Code

```
/*
The Wunderkammer Interface
a live processing platform for any instrumentation
by Benjamin J M Klein
©2017

Last Updated: 2/14/17
How to run the computer part
1. Make sure that your audio device of choice (audio interface, built-in microphone, etc.) is set as the default input and output for the computer.
2. Open Wunderkammer.scd in SuperCollider (Download at supercollider.github.io if SuperCollider is not installed already)
3. Go to Menu and select Language->Evaluate File, or select all (command A) and press enter.
4. The program interface will appear on your screen, follow the instructions on the interface to activate it.
5. Press command+period(.) or select Language->Stop to stop the computer part

For more information, feel free to contact Benjamin J M Klein at bjmklein@gmail.com
*/


s.waitForBoot{


var b1a, b2a, b3a, b4a, b5a, b6a, b7a, b8a, b9a, b10a,
   b11a, b12a, b13a, b14a, b15a, b16a, b17a, b18a, b19a, b20a,
   b21a, b22a, b23a, b24a, b25a,
   b1b, b2b, b3b, b4b, b5b, b6b, b7b, b8b, b9b, b10b,
   b11b, b12b, b13b, b14b, b15b, b16b, b17b, b18b, b19b, b20b,
   b21b, b22b, b23b, b24b, b25b,
   w1, w2, w3, w4, w5, w6, w7, w8, w9, w10,
   w11, w12, w13, w14, w15, w16, w17, w18, w19, w20,
   w21, w22, w23, w24, w25,
   sw1, sw2, sw3, sw4, sw5;

Server.default.makeGui;

(

//Amplification of the source sound

SynthDef("mic", { arg out=0, amp=0.05, pan=0;
        var audio;
                audio = (SoundIn.ar([0, 1]));
           Date.getDate.postln;
           SystemClock.sched(0.0,{ arg time; time.postln});
                Out.ar(out, Pan2.ar(audio, pan, amp))
        }).play;


// processing activated by button 1
```

```
(

SynthDef("wunderkammer1", { arg out=0, amp=0.5, pan=0, freq = 400, dur=58, legato = 61/58;
	var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen, reverb,
	   sinenv1, sinenvgen1, modulation, process, ampenv, ampenvgen, penv, penvgen;

	audio = (SoundIn.ar([0]));

	env1 = Env.new([2000, 2005, 16000, 8000, 2016, 2028.8],
			[25.3, 1.7, 3.5, 6.2, 23.5], 'linear');
	envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

	filter1 =  BLowPass.ar(audio, envgen1);

	env2 = Env.new([1980, 1973, 16, 360, 1932, 1825.6],
			[22.3, 2.7, 5.5, 8.2, 21.5], 'linear');
	envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

	filter2 = BHiPass.ar(filter1, envgen2);

	renv = Env.new([1, 0.96],
			[60.2], 'linear');
	renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

	reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

	sinenv1 = Env.new([452, 397, 563, 422, 151, 98, 121 ],
				[ 10.05, 10.05, 10.05, 0.1, 15.05, 15.05 ], 'sine');
	sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

	ampenv = Env.new([0, 0.7, 0.6, 0, 0, 0.4, 0.3, 0, 0, 0.6, 0.8, 0, 0, 0.4, 0.2, 0, 0, 0.5, 0.3, 0 ],
			[10.5, 1.7, 0.3, 2.3, 7.7, 2.8, 1.8, 1.1, 7.4, 0.4, 1.6, 3.3, 5.6, 1.4, 3.7, 3.4, 5.5, 1.7,
			8.5], 'linear');

	ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

	modulation = (SinOsc.ar(sinenvgen1, 0, ampenvgen));

	process = modulation * reverb;

	env3 = Env.new([0, 0.2, 0.4, 0.3, 0.5, 0.10128, 0],
			[0.1, 28, 15, 10, 7, 0.1], 'linear');
	envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

	penv = Env.new([0, -1, 0.3, -0.5, 0.6, 0, -0.2, -0.8, 0.5, 0],
				[ 17.5, 5, 4.5, 18.3, 15.7 ], 'linear');
	penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

	Out.ar(out, Pan2.ar(process, penvgen, envgen3))
		}).add;


~a = Buffer.alloc(s, 44100 * 3, 1, 0, 0);
```

```
SynthDef(\lowrec1, { arg bufnum = 0;
	var audio, rec;
	audio = (SoundIn.ar([0]));
	rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;

SynthDef (\lowplay1a, { arg out = 0, amp=0.3, pan= -0.55, bufnum = 0 ;
	var audio, reverb, env, envgen;
	audio = PlayBuf.ar(1, bufnum, 0.356);
	reverb = FreeVerb.ar(audio, 0.78, 0.78, 0.78);
	env = Env.new([0, 0.5, 0.5, 0],
			[0.2, 8.02, 0.2 ], 'linear');
	envgen = EnvGen.kr(env, 1.0, doneAction: 2);
	Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\lowplay1b, { arg out = 0, amp=0.3, pan= -0.35, bufnum = 0 ;
	var audio, reverb, env, envgen;
	audio = PlayBuf.ar(1, bufnum, 0.467);
	reverb = FreeVerb.ar(audio, 0.48, 0.48, 0.48);
	env = Env.new([0, 0.5, 0.5, 0],
			[0.2, 6.02, 0.2 ], 'linear');
	envgen = EnvGen.kr(env, 1.0, doneAction: 2);
	Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;


a = Pbind(*[
	instrument:	"wunderkammer1",
	freq:		Pseq([400], 1),
	dur:		Pseq([61], 1),
]);


c = Prout({

	0.1.wait;
	Synth(\lowrec1).postln;
	SystemClock.sched(0.0,{ arg time; time.postln});
	18.43.wait;
	Synth(\lowplay1a).postln;
	11.57.wait;
	16.43.wait;
	Synth(\lowplay1b).postln;
	20.wait;
	~a.freeMsg;
});

~button1 = Ppar([a, c], 1);

);


// end button 1
```

```
//processing activated by button 2
(

SynthDef("wunderkammer2", { arg out=0, amp=0.1, pan=0, freq = 400;
	var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen, reverb,
	sinenv1, sinenvgen1, sinenv2, sinenvgen2, sinenv3, sinenvgen3, sinenv4, sinenvgen4,
	   modulation, process, ampenv, ampenvgen,penv, penvgen;
	audio = (SoundIn.ar([0]));

	env1 = Env.new([2028.8, 2035, 7000, 3000, 2115.2],
			[24, 11, 8.2, 17], 'linear');
	envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

	filter1 =  BLowPass.ar(audio, envgen1);

	env2 = Env.new([1825.6, 612, 780, 300, 1200, 1677.4],
			[12, 23.1, 6, 6.1, 13], 'linear');
	envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

	filter2 = BHiPass.ar(filter1, envgen2);

	renv = Env.new([0.96, 0.92],
			[60.2], 'linear');
	renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

	reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

	sinenv1 = Env.new([2715, 2498, 2644, 2350,  2550, 1112, 989, 1266, 1018 ],
				[ 7.55, 7.55, 7.55, 7.55, 0.1, 10.05, 10.05, 10.05 ], 'sine');
	sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

	sinenv2 = Env.new([2164, 2200, 2000, 2137, 1112, 1000, 1200, 1018, 1209, 1018 ],
				[ 10.05, 10.05, 10.05, 0.1, 7.55, 7.55, 7.55, 7.55 ], 'sine');
	sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

	sinenv3 = Env.new([687, 600, 645, 480, 430, 476 ],
				[ 15.05, 15.05, 0.1, 15.05, 15.05 ], 'sine');
	sinenvgen3 = EnvGen.kr(sinenv3, 1.0, doneAction: 2);

	sinenv4 = Env.new([409, 437, 480, 476 ],
				[ 30.05, 0.1, 30.05 ], 'sine');
	sinenvgen4 = EnvGen.kr(sinenv4, 1.0, doneAction: 2);

	ampenv = Env.new([0, 0.3, 0.6, 0.2, 0.1,    0.4, 0.3,    0.1, 0.2,   0.8, 0.5, 0.3,  0.6, 0.4, 0.6,   0.2,
				0.1,   0.5, 0.3,   0 ],
			[   16.5, 1.7,  6.3,  6.3, 3.7,  2.8, 1.8, 4.1, 2.4,  0.4, 1.6,   12.3, 1.6, 1.4, 6.7,
				3.4, 2.5, 1.7, 3.5], 'linear');
	ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

	modulation = (SinOsc.ar([sinenvgen1,sinenvgen2, sinenvgen3, sinenvgen4], 0, ampenvgen));

	process = modulation * reverb;

	penv = Env.new([0, -0.3, 1, -0.5, 0, 0.4, -0.5, 0.3, 0.8, 0],
```

```
                                    [ 2.5, 18, 9.5, 5.3, 25.7 ], 'linear');
        penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

        env3 = Env.new([0, 0.55128, 0.55512, 0],
                        [0.1, 60, 0.1], 'linear');
        envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

        Out.ar(out, Pan2.ar(process, penvgen, envgen3))
            }).add;


~b = Buffer.alloc(s, 44100 * 10, 1, 0, 3);


SynthDef(\highrec2, { arg bufnum = 3;
        var audio, rec;
        audio = (SoundIn.ar([0]));
        rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;

SynthDef (\highplay2a, { arg out = 0, amp=0.3, pan=0.35, bufnum = 3 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 1.556);
        reverb = FreeVerb.ar(audio, 0.52, 0.52, 0.52);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 6.02, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\highplay2b, { arg out = 0, amp=0.3, pan= -0.06, bufnum = 3 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 1.06);
        reverb = FreeVerb.ar(audio, 0.1, 0.1, 0.1);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 9.03, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;




a = Pbind(*[
        instrument:     "wunderkammer2",
        freq:           Pseq([400], 1),
        dur:            Pseq([61], 1),
]);


d = Prout({

        0.1.wait;
        Synth(\highrec2).postln;
        SystemClock.sched(0.0,{ arg time; time.postln});
```

```
        18.67.wait;
        Synth(\highplay2a).postln;
        11.33.wait;
        11.39.wait;
        Synth(\highplay2b).postln;
        20.wait;
        ~b.freeMsg;
});

~button2 = Ppar([a, d], 1);

);


// end button 2


//processing activated by button 3
(

SynthDef("wunderkammer3", { arg out=0, amp=0.1, pan=0, freq = 400;
                var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                        reverb,
            sinenv1, sinenvgen1, sinenv2, sinenvgen2, sinenv3, sinenvgen3, sinenv4, sinenvgen4,
            modulation, process, ampenv, ampenvgen, penv, penvgen;
        audio = (SoundIn.ar([0]));

        env1 = Env.new([2115.2, 2180, 3000, 12000, 6000, 2306, 2259.2],
                        [10, 6, 24, 7, 0.2, 13], 'linear');
        envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

        filter1 =  BLowPass.ar(audio, envgen1);

        env2 = Env.new([1677.4, 1535.6],
                        [60.2], 'linear');
        envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

        filter2 = BHiPass.ar(filter1, envgen2);

        renv = Env.new([0.92, 0.88],
                        [60.2], 'linear');
        renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

        reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

        sinenv1 = Env.new([829, 735, 4526, 4361],
                                [ 30.05, 0.1, 30.05 ], 'sine');
        sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

        sinenv2 = Env.new([829, 967, 735, 2459, 2240, 2678, 2389, 2432 ],
                                [ 15.05, 15.05, 0.1, 7.55, 7.55, 7.55, 7.55 ], 'sine');
        sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

        sinenv3 = Env.new([468, 440, 480, 464, 1145, 1000, 1150, 1103 ],
                                [ 10.05, 10.05, 10.05, 0.1, 10.05, 10.05, 10.05 ], 'sine');
```

```
        sinenvgen3 = EnvGen.kr(sinenv3, 1.0, doneAction: 2);

        sinenv4 = Env.new([468, 498, 439, 486, 464, 103, 142,  131 ],
                                [ 7.55, 7.55, 7.55, 7.55, 0.1, 15.05, 15.05 ], 'sine');
        sinenvgen4 = EnvGen.kr(sinenv4, 1.0, doneAction: 2);

        ampenv = Env.new([0, 0.2,   0.3,    0.7, 0.2,    0.2, 0.3,    0, 0.2,   0.6, 0.8, 0.3,  0.5, 0.4, 0.6,   0.2,
                                0.6,   0.5, 0.3,   0 ],
                            [   5.5, 1.7,  0.3,  6.3, 3.7,  2.8, 3.8, 4.1, 2.4,  3.4, 1.6,   4.3, 1.6, 4.4, 6.7, 3.4,
                                2.5, 1.7, 8.5], 'linear');
        ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

        modulation = (SinOsc.ar([sinenvgen1,sinenvgen2, sinenvgen3, sinenvgen4], 0, ampenvgen));

        process = modulation * reverb;

        penv = Env.new([0, 0.5, -0.4, 0.2, -0.7, 0.6, -0.2, -0.8, -0.5, 0],
                            [ 8.5, 19, 9.5, 18.3, 5.7 ], 'linear');
        penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

        env3 = Env.new([0, 0.60512, 0.61152, 0],
                        [0.1, 60, 0.1], 'linear');
        envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

        Out.ar(out, Pan2.ar(process, penvgen, envgen3))
            }).add;



~c = Buffer.alloc(s, 44100 * 3, 1, 0, 4);

SynthDef(\lowrec3, { arg bufnum = 4;
        var audio, rec;
        audio = (SoundIn.ar([0]));
        rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;



SynthDef (\lowplay3a, { arg out = 0, amp=0.3, pan= -0.5, bufnum = 4 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 0.381);
        reverb = FreeVerb.ar(audio, 0.7, 0.7, 0.7);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 7.47, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\lowplay3b, { arg out = 0, amp=0.3, pan= -0.47, bufnum = 4 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 0.396);
        reverb = FreeVerb.ar(audio, 0.66, 0.66, 0.66);
        env = Env.new([0, 0.5, 0.5, 0],
```

```
                        [0.2, 7.17, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;


a = Pbind(*[
        instrument:     "wunderkammer3",
        freq:           Pseq([400], 1),
        dur:            Pseq([61], 1),
]);


c = Prout({

        0.5.wait;
        Synth(\lowrec3).postln;
        SystemClock.sched(0.0,{ arg time; time.postln});
        17.86.wait;
        Synth(\lowplay3a).postln;
        12.14.wait;
        17.57.wait;
        Synth(\lowplay3b).postln;
        20.wait;
        ~c.freeMsg

});

~button3 = Ppar([a, c], 1);

);


// end button 3


//processing activated by button 4
(

SynthDef("wunderkammer4", { arg out=0, amp=0.1, pan=0, freq = 400;
                var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                        reverb,
            sinenv1, sinenvgen1, sinenv2, sinenvgen2, sinenv3, sinenvgen3, sinenv4, sinenvgen4,
            modulation, process, ampenv, ampenvgen, penv, penvgen;
        audio = (SoundIn.ar([0]));

        env1 = Env.new([2259.2, 2460.8],
                        [60.2], 'linear');
        envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

        filter1 =  BLowPass.ar(audio, envgen1);

        env2 = Env.new([1535.6, 400, 1700, 200, 317, 2015, 1400],
                        [20, 3, 14, 7, 14, 17.2], 'linear');
        envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);
```

```
        filter2 = BHiPass.ar(filter1, envgen2);

        renv = Env.new([0.88, 0.84],
                        [60.2], 'linear');
        renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

        reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

        sinenv1 = Env.new([1679, 1486, 1786, 1584, 2220, 1800, 2409, 1769, 2056],
                                [ 10.05, 10.05, 10.05, 0.1, 7.55, 7.55, 7.55, 7.55 ], 'sine');
        sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

        sinenv2 = Env.new([1679, 1584, 2043, 2387, 2056 ],
                                [ 30.05, 0.1, 15.05, 15.05 ], 'sine');
        sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

        sinenv3 = Env.new([504, 450, 500, 562, 496, 678, 520 ],
                                [ 15.05, 15.05, 0.1, 10.05, 10.05, 10.05 ], 'sine');
        sinenvgen3 = EnvGen.kr(sinenv3, 1.0, doneAction: 2);

        sinenv4 = Env.new([504, 500, 492, 520 ],
                                [ 30.05, 0.1, 30.05 ], 'sine');
        sinenvgen4 = EnvGen.kr(sinenv4, 1.0, doneAction: 2);

        ampenv = Env.new([0, 0.7,   0.6,    0.3, 0.4,    0.4, 0.3,    0.1, 0.6,   0.6, 0.1, 0,  0.6, 0.4, 0.3,   0.1,
                                0.5,   0.5, 0.3,   0 ],
                            [   17.5, 1.7,  0.3,  4.3, 2.7,  1.8, 1.8, 3.1, 1.4,  0.4, 1.6,   7.3, 1.6, 1.4, 6.7,
                                2.4, 2.5, 1.7, 8.5], 'linear');
        ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

        modulation = (SinOsc.ar([sinenvgen1,sinenvgen2, sinenvgen3, sinenvgen4], 0, ampenvgen));
        process = reverb * modulation;

        penv = Env.new([0, -0.5, -1, 0.5, 0.3, 0, -0.2, 0.8, 0.5, 0],
                            [ 12.5, 15, 9.5, 8.3, 15.7 ], 'linear');
        penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

        env3 = Env.new([0, 0.61152, 0.62048, 0],
                        [60.2], 'linear');
        envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

        Out.ar(out, Pan2.ar(process, penvgen, envgen3))
            }).add;


~d = Buffer.alloc(s, 44100 * 10, 1, 0, 7);


SynthDef(\highrec4, { arg bufnum = 7;
        var audio, rec;
        audio = (SoundIn.ar([0]));
        rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;
```

```
SynthDef (\highplay4a, { arg out = 0, amp=0.3, pan=0.32, bufnum = 7 ;
	var audio, reverb, env, envgen;
	audio = PlayBuf.ar(1, bufnum, 1.489);
	reverb = FreeVerb.ar(audio, 0.48, 0.48, 0.48);
	env = Env.new([0, 0.5, 0.5, 0],
			[0.2, 6.31, 0.2 ], 'linear');
	envgen = EnvGen.kr(env, 1.0, doneAction: 2);
	Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\highplay4b, { arg out = 0, amp=0.3, pan= -0.52, bufnum = 7 ;
	var audio, reverb, env, envgen;
	audio = PlayBuf.ar(1, bufnum, 2.059);
	reverb = FreeVerb.ar(audio, 0.74, 0.74, 0.74);
	env = Env.new([0, 0.5, 0.5, 0],
			[0.2, 4.45, 0.2 ], 'linear');
	envgen = EnvGen.kr(env, 1.0, doneAction: 2);
	Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;


a = Pbind(*[
	instrument:	"wunderkammer4",
	freq:		Pseq([400], 1),
	dur:		Pseq([61], 1),
]);

d = Prout({

	0.1.wait;
	Synth(\highrec4).postln;
	SystemClock.sched(0.0,{ arg time; time.postln});
	17.98.wait;
	Synth(\highplay4a).postln;
	12.02.wait;
	22.49.wait;
	Synth(\highplay4b).postln;
	20.wait;
	~d.freeMsg

});

~button4 = Ppar([a, d], 1);

);


// end button 4



//processing activated by button 5
```

```
(

SynthDef("wunderkammer5", { arg out=0, amp=0.1, pan=0, freq = 400;
                var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                        reverb,
            sinenv1, sinenvgen1, sinenv2, sinenvgen2, sinenv3, sinenvgen3, sinenv4, sinenvgen4,
            modulation, process, ampenv, ampenvgen, penv, penvgen;
        audio = (SoundIn.ar([0]));

        env1 = Env.new([2460.8, 2005, 6000, 18000, 2016, 2720],
                        [7.3, 10.7, 10.5, 7.2, 22.5], 'linear');
        envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

        filter1 =  BLowPass.ar(audio, envgen1);

        env2 = Env.new([1400, 1112, 1780, 1300, 200, 1270.8],
                        [10, 3.1, 8, 26.1, 13], 'linear');
        envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

        filter2 = BHiPass.ar(filter1, envgen2);

        renv = Env.new([0.84, 0.8],
                        [60.2], 'linear');
        renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

        reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

        sinenv1 = Env.new([1773, 1533, 1895, 1603, 1679, 2385, 1875, 2574, 2220 ],
                                [ 7.55, 7.55, 7.55, 7.55, 0.1, 10.05, 10.05, 10.05 ], 'sine');
        sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

        sinenv2 = Env.new([1773, 1356, 1679, 2110, 1933, 2043 ],
                                [ 15.05, 15.05, 0.1, 15.05, 15.05 ], 'sine');
        sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

        sinenv3 = Env.new([508, 347, 765,  504, 603,  489, 780, 562 ],
                                [ 10.05, 10.05, 10.05, 0.1, 10.05, 10.05, 10.05 ], 'sine');
        sinenvgen3 = EnvGen.kr(sinenv3, 1.0, doneAction: 2);

        sinenv4 = Env.new([508, 786, 432, 504, 464, 376, 578, 400, 492 ],
                                [ 10.05, 10.05, 10.05, 0.1, 7.55, 7.55, 7.55, 7.55 ], 'sine');
        sinenvgen4 = EnvGen.kr(sinenv4, 1.0, doneAction: 2);

        ampenv = Env.new([0, 0.5,   0.7,    0.5, 2,    0.4, 0.3,    0, 0.5,   0.3, 0.6, 0.2,  0.1, 0.4, 0.7,   0.4, 0,
                                0.5, 0.3,   0 ],
                        [   1.5, 1.7,  2.3,  8.3, 3.7,  4.8, 1.8, 4.1, 2.4,  1.4, 3.6,   1.3, 1.6, 1.4, 6.7, 5.4,
                                2.5, 1.7, 12.5], 'linear');
        ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

        modulation = (SinOsc.ar([sinenvgen1,sinenvgen2, sinenvgen3, sinenvgen4], 0, ampenvgen));
        process = modulation * reverb;

        env3 = Env.new([0, 0.62048, 0.632, 0],
                        [0.1, 60, 0.1], 'linear');
        envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);
```

```
        penv = Env.new([0, 0.5, 1, 0.5, -0.3, 0, -0.2, -0.8, 0.5, 0],
                                [ 12.5, 5, 19.5, 18.3, 5.7 ], 'linear');
        penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

        Out.ar(out, Pan2.ar(process, penvgen, envgen3))
            }).add;


~e = Buffer.alloc(s, 44100 * 3, 1, 0, 8);

SynthDef(\lowrec5, { arg bufnum = 8;
        var audio, rec;
        audio = (SoundIn.ar([0]));
        rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;

SynthDef (\lowplay5a, { arg out = 0, amp=0.3, pan= -0.26, bufnum = 8 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 0.538);
        reverb = FreeVerb.ar(audio, 0.36, 0.36, 0.36);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 5.17, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\lowplay5b, { arg out = 0, amp=0.3, pan= 0.18, bufnum = 8 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 0.618);
        reverb = FreeVerb.ar(audio, 0.26, 0.26, 0.26);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 4.45, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;


a = Pbind(*[
        instrument:     "wunderkammer5",
        freq:           Pseq([400], 1),
        dur:            Pseq([61], 1),
]);


c = Prout({

        0.1.wait;
        Synth(\lowrec5).postln;
        SystemClock.sched(0.0,{ arg time; time.postln});
        15.57.wait;
        Synth(\lowplay5a).postln;
        14.43.wait;
        14.86.wait;
        Synth(\lowplay5b).postln;
```

```
        20.wait;
        ~e.freeMsg
});


~button5 = Ppar([a, c], 1);

);


// end button 5


//processing activated by button 6
(

SynthDef("wunderkammer6", { arg out=0, amp=0.1, pan=0, freq = 400;
                var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                        reverb,
            sinenv1, sinenvgen1, sinenv2, sinenvgen2,
            modulation, process, ampenv, ampenvgen, penv, penvgen;

        audio = (SoundIn.ar([0]));

        env1 = Env.new([2720, 2180, 13000, 12000, 6000, 2306, 3063.8],
                        [1, 15, 4, 17, 20.2, 3], 'linear');
        envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

        filter1 =  BLowPass.ar(audio, envgen1);

        env2 = Env.new([1270.8, 2112, 170, 300, 1200, 1147.9],
                        [17, 13.1, 1, 16.1, 13], 'linear');
        envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

        filter2 = BHiPass.ar(filter1, envgen2);

        renv = Env.new([0.8, 0.76],
                        [60.2], 'linear');
        renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

        reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

        sinenv1 = Env.new([1301, 1207, 422, 392 ],
                                [ 30.05, 0.1, 30.05 ], 'sine');
        sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

        sinenv2 = Env.new([488, 367, 675, 278, 484, 422, 435, 489, 392 ],
                                [ 7.55, 7.55, 7.55, 7.55, 0.1, 10.05, 10.05, 10.05 ], 'sine');
        sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

        ampenv = Env.new([0, 0.2,   0.6,    0.2, 0.5,    0.2, 0.4,    0.1, 0.1,   0.6, 0.3, 0.4,  0.1, 0.7, 0.2,   0,
                                0.6,   0.5, 0,   0 ],
                            [   3.5, 1.7,  3.3,  6.3, 3.7,  6.8, 0.8, 4.1, 7.4,  0.4, 1.6,   3.3, 2.6, 2.4, 4.7, 3.4,
                                2.5, 1.7, 8.5], 'linear');
        ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);
```

```
        modulation = (SinOsc.ar([sinenvgen1,sinenvgen2], 0, ampenvgen));
        process = modulation * reverb;

        penv = Env.new([0, -0.1, 0.1, -0.5, 0.4, 0, -0.2, 0, 0.7, 0],
                                [ 10.5, 16, 8.5, 8.3, 17.7 ], 'linear');
        penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

        env3 = Env.new([0, 0.652, 0.65608, 0],
                        [0.1, 60, 0.1], 'linear');
        envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

        Out.ar(out, Pan2.ar(process, penvgen, envgen3))
            }).add;

~f = Buffer.alloc(s, 44100 * 10, 1, 0, 11);

SynthDef(\highrec6, { arg bufnum = 11;
        var audio, rec;
        audio = (SoundIn.ar([0]));
        rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;


SynthDef (\highplay6a, { arg out = 0, amp=0.3, pan=0.23, bufnum = 11 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 1.321);
        reverb = FreeVerb.ar(audio, 0.36, 0.36, 0.36);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 7.17, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\highplay6b, { arg out = 0, amp=0.3, pan= 0.06, bufnum = 11 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 1.077);
        reverb = FreeVerb.ar(audio, 0.12, 0.12, 0.12);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 8.88, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;



a = Pbind(*[
        instrument:     "wunderkammer6",
        freq:           Pseq([400], 1),
        dur:            Pseq([61], 1),
]);


d = Prout({
```

```
		0.1.wait;
		Synth(\highrec6).postln;
		SystemClock.sched(0.0,{ arg time; time.postln});
		15.9.wait;
		Synth(\highplay6a).postln;
		14.1.wait;
		11.73.wait;
		Synth(\highplay6b).postln;
		20.wait;
		~f.freeMsg
});

~button6 = Ppar([a, d], 1);

);


// end button 6


//processing activated by button 7
(

SynthDef("wunderkammer7", { arg out=0, amp=0.1, pan=0, freq = 400;
			var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
				reverb,
		sinenv1, sinenvgen1, sinenv2, sinenvgen2, sinenv3, sinenvgen3, sinenv4, sinenvgen4,
		modulation, process, ampenv, ampenvgen, penv, penvgen;

		audio = (SoundIn.ar([0]));

		env1 = Env.new([3063.8, 3411.2],
				[60.2], 'linear');
		envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

		filter1 =  BLowPass.ar(audio, envgen1);

		env2 = Env.new([1147.9, 1031.2],
				[60.2], 'linear');
		envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

		filter2 = BHiPass.ar(filter1, envgen2);

		renv = Env.new([0.76, 0.72],
				[60.2], 'linear');
		renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

		reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

		sinenv1 = Env.new([241, 136, 211, 3703, 4576, 3538 ],
					[ 15.05, 15.05,  0.1, 15.05, 15.05  ], 'sine');
		sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

		sinenv2 = Env.new([241, 315, 169, 211, 2325, 1567, 3289, 1796, 2298 ],
```

```
                                    [ 10.05, 10.05, 10.05, 0.1, 7.55, 7.55, 7.55, 7.55  ], 'sine');
        sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

        sinenv3 = Env.new([241, 211, 937, 689, 1037,  895 ],
                                    [ 30.05, 0.1, 10.05, 10.05, 10.05 ], 'sine');
        sinenvgen3 = EnvGen.kr(sinenv3, 1.0, doneAction: 2);

        sinenv4 = Env.new([241, 211, 242, 270 ],
                                    [ 30.05, 0.1, 30.05 ], 'sine');
        sinenvgen4 = EnvGen.kr(sinenv4, 1.0, doneAction: 2);

        ampenv = Env.new([0, 0.2,   0.3,    0.7, 0.3,    0.5, 0.6,    0.2, 0.4,   0.1, 0.8, 0.1,  0.2, 0.4, 0.7,   0.3,
                                        0,   0.2, 0.6,   0 ],
                                [   5.5, 1.7,  0.3,  9.3, 3.7,  4.8, 1.8, 4.1, 2.4,  2.4, 1.6,   9.3, 1.6, 1.4, 6.7, 3.4,
                                        2.5, 1.7, 4.5], 'linear');
        ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

        modulation = (SinOsc.ar([sinenvgen1,sinenvgen2, sinenvgen3, sinenvgen4], 0, ampenvgen));
        process = modulation * reverb;

        env3 = Env.new([0, 0.66608, 0.66272, 0],
                            [0.1, 60, 0.1], 'linear');
        envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

        penv = Env.new([0, 0.1, -0.4, 0.1, 0.3, 0, -0.8, 0.8, 0, 0],
                                [ 17.5, 10, 14.5, 8.3, 10.7 ], 'linear');
        penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

        Out.ar(out, Pan2.ar(process, penvgen, envgen3))
            }).add;


~g = Buffer.alloc(s, 44100 * 3, 1, 0, 12);

SynthDef(\lowrec7, { arg bufnum = 12;
        var audio, rec;
        audio = (SoundIn.ar([0]));
        rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;


SynthDef (\lowplay7a, { arg out = 0, amp=0.3, pan=0.61, bufnum = 12 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 0.328);
        reverb = FreeVerb.ar(audio, 0.88, 0.88, 0.88);
        env = Env.new([0, 0.5, 0.5, 0],
                            [0.2, 9.74, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\lowplay7b, { arg out = 0, amp=0.3, pan= 0.2, bufnum = 12 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 0.583);
        reverb = FreeVerb.ar(audio, 0.3, 0.3, 0.3);
```

```
		env = Env.new([0, 0.5, 0.5, 0],
				[0.2, 4.74, 0.2 ], 'linear');
		envgen = EnvGen.kr(env, 1.0, doneAction: 2);
		Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

a = Pbind(*[
		instrument:		"wunderkammer7",
		freq:			Pseq([400], 1),
		dur:		Pseq([61], 1),
]);



c = Prout({

		0.1.wait;
		Synth(\lowrec7).postln;
		SystemClock.sched(0.0,{ arg time; time.postln});
		19.14.wait;
		Synth(\lowplay7a).postln;
		10.86.wait;
		15.14.wait;
		Synth(\lowplay7b).postln;
		20.wait;
		~g.freeMsg
});


~button7 = Ppar([a, c], 1);

);


// end button 7



//processing activated by button 8
(

SynthDef("wunderkammer8", { arg out=0, amp=0.1, pan=0, freq = 400;
			var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
				reverb,
		sinenv1, sinenvgen1, sinenv2, sinenvgen2, sinenv3, sinenvgen3, sinenv4, sinenvgen4,
		modulation, process, ampenv, ampenvgen, penv, penvgen;
		audio = (SoundIn.ar([0]));

		env1 = Env.new([3411.2, 2780, 3000, 12000, 16000, 2306, 3843.2],
				[8, 7, 4, 17, 3.2, 21], 'linear');
		envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

		filter1 =  BLowPass.ar(audio, envgen1);

		env2 = Env.new([1031.2, 920.9],
```

```
                        [60.2], 'linear');
        envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

        filter2 = BHiPass.ar(filter1, envgen2);

        renv = Env.new([0.72, 0.68],
                        [60.2], 'linear');
        renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

        reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

        sinenv1 = Env.new([2550,  3684, 2385, 3867, 3703 ],
                                [ 15.05, 15.05,  0.1, 30.05 ], 'sine');
        sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

        sinenv2 = Env.new([2137, 1950, 3498, 1089, 2110, 2352, 1374, 2325 ],
                                [ 7.55, 7.55, 7.55, 7.55, 0.1, 15.05, 15.05 ], 'sine');
        sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

        sinenv3 = Env.new([645, 473, 893,  603, 978, 684, 1204, 937 ],
                                [ 10.05, 10.05, 10.05, 0.1, 10.05, 10.05, 10.05 ], 'sine');
        sinenvgen3 = EnvGen.kr(sinenv3, 1.0, doneAction: 2);

        sinenv4 = Env.new([437, 464, 214, 134, 328, 185, 242 ],
                                [ 30.05, 0.1, 7.55, 7.55, 7.55, 7.55 ], 'sine');
        sinenvgen4 = EnvGen.kr(sinenv4, 1.0, doneAction: 2);

        ampenv = Env.new([0, 0.7,   0.2,    0.6, 0.1,    0.2, 0.3,    0.3, 0.4,   0.3, 0.8, 0.5,  0, 0.4, 0.2,   0,
                                0.7,   0.5, 0.6,   0 ],
                        [   5.5, 6.7,  2.3,  4.3, 3.7,  2.8, 1.8, 1.1, 5.4,  0.4, 1.6,   7.3, 1.6, 1.4, 6.7, 3.4,
                                2.5, 1.7, 8.5], 'linear');
        ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

        modulation = (SinOsc.ar([sinenvgen1,sinenvgen2, sinenvgen3, sinenvgen4], 0, ampenvgen));
        process = modulation * reverb;

        penv = Env.new([0, 0.4, -0.3, 0, 0.8, 0.4, -0.7, 0.2, -0.1, 0],
                        [ 7.5, 20, 12.5, 2.3, 18.7 ], 'linear');
        penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

        env3 = Env.new([0, 0.67272, 0.67192, 0],
                        [0.1, 60, 0.1], 'linear');
        envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

        Out.ar(out, Pan2.ar(process, penvgen, envgen3))
                }).add;

~h = Buffer.alloc(s, 44100 * 10, 1, 0, 15);

SynthDef(\highrec8, { arg bufnum = 15;
        var audio, rec;
        audio = (SoundIn.ar([0]));
        rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;
```

```
SynthDef (\highplay8a, { arg out = 0, amp=0.3, pan=0.09, bufnum = 15 ;
	var audio, reverb, env, envgen;
	audio = PlayBuf.ar(1, bufnum, 1.111);
	reverb = FreeVerb.ar(audio, 0.16, 0.16, 0.16);
	env = Env.new([0, 0.5, 0.5, 0],
			[0.2, 8.6, 0.2 ], 'linear');
	envgen = EnvGen.kr(env, 1.0, doneAction: 2);
	Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\highplay8b, { arg out = 0, amp=0.3, pan= 0.52, bufnum = 15 ;
	var audio, reverb, env, envgen;
	audio = PlayBuf.ar(1, bufnum, 2.121);
	reverb = FreeVerb.ar(audio, 0.76, 0.76, 0.76);
	env = Env.new([0, 0.5, 0.5, 0],
			[0.2, 4.41, 0.2 ], 'linear');
	envgen = EnvGen.kr(env, 1.0, doneAction: 2);
	Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;




a = Pbind(*[
	instrument:	"wunderkammer8",
	freq:		Pseq([400], 1),
	dur:		Pseq([61], 1),
]);

d = Prout({

	0.1.wait;
	Synth(\highrec8).postln;
	SystemClock.sched(0.0,{ arg time; time.postln});
	12.43.wait;
	Synth(\highplay8a).postln;
	17.57.wait;
	22.84.wait;
	Synth(\highplay8b).postln;
	20.wait;
	~h.freeMsg

});

~button8 = Ppar([a, d], 1);

);


// end button 8


//processing activated by button 9
```

```
(

SynthDef("wunderkammer9", { arg out=0, amp=0.1, pan=0, freq = 400;
                var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                                reverb,
        sinenv1, sinenvgen1, sinenv2, sinenvgen2, sinenv3, sinenvgen3, sinenv4, sinenvgen4,
        modulation, process, ampenv, ampenvgen, penv, penvgen;
        audio = (SoundIn.ar([0]));

        env1 = Env.new([3843.2, 4332.8],
                        [60.2], 'linear');
        envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

        filter1 =  BLowPass.ar(audio, envgen1);

        env2 = Env.new([920.9, 212, 1170, 900, 100, 816.9],
                        [27, 3.1, 11, 6.1, 13], 'linear');
        envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

        filter2 = BHiPass.ar(filter1, envgen2);

        renv = Env.new([0.68, 0.64],
                        [60.2], 'linear');
        renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

        reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

        sinenv1 = Env.new([181, 78, 377,  151, 4691, 8933, 3867,  4526 ],
                                [ 10.05, 10.05, 10.05, 0.1, 10.05, 10.05, 10.05], 'sine');
        sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

        sinenv2 = Env.new([181, 375, 87, 238, 151, 2486, 2459 ],
                                [ 7.55, 7.55, 7.55, 7.55, 0.1, 30.05 ], 'sine');
        sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

        sinenv3 = Env.new([181,  278, 151, 1187, 2376, 1145 ],
                                [ 15.05, 15.05, 0.1, 15.05, 15.05 ], 'sine');
        sinenvgen3 = EnvGen.kr(sinenv3, 1.0, doneAction: 2);

        sinenv4 = Env.new([181, 345, 39, 289, 151, 76, 154, 57, 188, 103 ],
                                [ 7.55, 7.55, 7.55, 7.55,  0.1, 7.55, 7.55, 7.55, 7.55 ], 'sine');
        sinenvgen4 = EnvGen.kr(sinenv4, 1.0, doneAction: 2);

        ampenv = Env.new([0, 0.3,   0.5,    0, 0.7,    0, 0.3,    0, 0.3,   0.6, 0.3, 0.7,  0, 0.2, 0.7,   0, 0.2,
                                0.5, 0.5,   0 ],
                        [   8.5, 0.7,  0.3,  6.3, 3.7,  2.8, 1.8, 4.1, 1.4,  0.4, 1.6,   5.3, 1.6, 1.4, 4.7, 2.4,
                                2.5, 10.7, 8.5], 'linear');
        ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

        modulation = (SinOsc.ar([sinenvgen1,sinenvgen2, sinenvgen3, sinenvgen4], 0, ampenvgen));
        process = modulation * reverb;

        penv = Env.new([0, 0.3, -0.7, 0, 0.3, -0.6, -0.2, -0.8, 0.7, 0],
                        [ 11.5, 16, 8.5, 10.3, 14.7 ], 'linear');
        penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);
```

```
		env3 = Env.new([0, 0.68192, 0.68368, 0],
				[0.1, 60, 0.1], 'linear');
		envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

		Out.ar(out, Pan2.ar(process, penvgen, envgen3))
			}).add;


~i = Buffer.alloc(s, 44100 * 3, 1, 0, 16);

SynthDef(\lowrec9, { arg bufnum = 16;
		var audio, rec;
		audio = (SoundIn.ar([0]));
		rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;


SynthDef (\lowplay9a, { arg out = 0, amp=0.3, pan= -0.64, bufnum = 16 ;
		var audio, reverb, env, envgen;
		audio = PlayBuf.ar(1, bufnum, 0.323);
		reverb = FreeVerb.ar(audio, 0.9, 0.9, 0.9);
		env = Env.new([0, 0.5, 0.5, 0],
				[0.2, 8.88, 0.2 ], 'linear');
		envgen = EnvGen.kr(env, 1.0, doneAction: 2);
		Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\lowplay9b, { arg out = 0, amp=0.3, pan= -0.44, bufnum = 16 ;
		var audio, reverb, env, envgen;
		audio = PlayBuf.ar(1, bufnum, 0.412);
		reverb = FreeVerb.ar(audio, 0.62, 0.62, 0.62);
		env = Env.new([0, 0.5, 0.5, 0],
				[0.2, 6.88, 0.2 ], 'linear');
		envgen = EnvGen.kr(env, 1.0, doneAction: 2);
		Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;


a = Pbind(*[
		instrument:		"wunderkammer9",
		freq:			Pseq([400], 1),
		dur:			Pseq([61], 1),
]);


c = Prout({

		0.1.wait;
		Synth(\lowrec9).postln;
		SystemClock.sched(0.0,{ arg time; time.postln});
		19.29.wait;
		Synth(\lowplay9a).postln;
		10.71.wait;
		17.29.wait;
```

```
        Synth(\lowplay9b).postln;
        20.wait;
        ~i.freeMsg

});


~button9 = Ppar([a, c], 1);

);


// end button 9


//processing activated by button 10
(

SynthDef("wunderkammer10", { arg out=0, amp=0.1, pan=0, freq = 400;
                var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                        reverb,
            sinenv1, sinenvgen1, sinenv2, sinenvgen2, sinenv3, sinenvgen3, sinenv4, sinenvgen4,
            modulation, process, ampenv, ampenvgen, penv, penvgen;
        audio = (SoundIn.ar([0]));

        env1 = Env.new([4332.8, 4880],
                        [60.2], 'linear');
        envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

        filter1 =  BLowPass.ar(audio, envgen1);

        env2 = Env.new([816.9, 719.2],
                        [60.2], 'linear');
        envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

        filter2 = BHiPass.ar(filter1, envgen2);

        renv = Env.new([0.64, 0.6],
                        [60.2], 'linear');
        renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

        reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

        sinenv1 = Env.new([4197, 3086, 5749, 3274, 4032, 60,  43, 89, 23, 30 ],
                                [ 7.55, 7.55, 7.55, 7.55, 0.1, 7.55, 7.55, 7.55, 7.55 ], 'sine');
        sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

        sinenv2 = Env.new([2406, 1389, 2743, 2379, 60, 75, 30 ],
                                [ 10.05, 10.05, 10.05,  0.1, 15.05, 15.05 ], 'sine');
        sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

        sinenv3 = Env.new([1062, 679, 1020, 60, 48, 76, 30 ],
                                [ 15.05, 15.05, 0.1, 10.05, 10.05, 10.05], 'sine');
        sinenvgen3 = EnvGen.kr(sinenv3, 1.0, doneAction: 2);
```

```
        sinenv4 = Env.new([159, 75, 235, 187, 60, 15, 93, 30 ],
                                [ 10.05, 10.05, 10.05, 0.1, 10.05, 10.05, 10.05 ], 'sine');
        sinenvgen4 = EnvGen.kr(sinenv4, 1.0, doneAction: 2);

        ampenv = Env.new([0, 0.7,  0.3,  0.2, 0.7,  0.2, 0.3,  0.1, 0.3,  0.7, 0.4, 0.2,  0, 0.8, 0.6,  0,
                                0.2,  0.5, 0.3,  0 ],
                            [  10.5, 1.7,  0.3,  4.3, 3.7,  0.8, 1.8, 10.1, 1.4,  2.4, 0.6,  9.3, 2.6, 1.4, 7.7,
                                3.4, 3.5, 1.7, 1.5], 'linear');
        ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

        modulation = (SinOsc.ar([sinenvgen1,sinenvgen2, sinenvgen3, sinenvgen4], 0, ampenvgen));
        process = modulation * reverb;

        penv = Env.new([0, 0.5, 0.3, -0.5, 0.3, 0.3, -0.2, 0.3, 0.4, 0],
                            [ 5.5, 10, 16.5, 15.3, 13.7 ], 'linear');
        penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

        env3 = Env.new([0, 0.69368, 0.698, 0],
                        [0.1, 60, 0.1], 'linear');
        envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

        Out.ar(out, Pan2.ar(process, penvgen, envgen3))
            }).add;

~j = Buffer.alloc(s, 44100 * 10, 1, 0, 19);

SynthDef(\highrec10, { arg bufnum = 19;
        var audio, rec;
        audio = (SoundIn.ar([0]));
        rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;


SynthDef (\highplay10a, { arg out = 0, amp=0.3, pan=0.67, bufnum = 19 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 3.043);
        reverb = FreeVerb.ar(audio, 0.96, 0.96, 0.96);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 2.88, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\highplay10b, { arg out = 0, amp=0.3, pan= -0.18, bufnum = 19 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 1.207);
        reverb = FreeVerb.ar(audio, 0.26, 0.26, 0.26);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 7.88, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;


a = Pbind(*[
```

```
        instrument:     "wunderkammer10",
        freq:           Pseq([400], 1),
        dur:            Pseq([61], 1),
]);

d = Prout({

        0.1.wait;
        Synth(\highrec10).postln;
        SystemClock.sched(0.0,{ arg time; time.postln});
        26.31.wait;
        Synth(\highplay10a).postln;
        3.69.wait;
        14.16.wait;
        Synth(\highplay10b).postln;
        20.wait;
        ~j.freeMsg

});

~button10 = Ppar([a, d], 1);

);


// end button 10


//processing activated by button 11
(

SynthDef("wunderkammer11", { arg out=0, amp=0.1, pan=0, freq = 400;
                var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                        reverb,
            sinenv1, sinenvgen1, sinenv2, sinenvgen2,
            modulation, process, ampenv, ampenvgen, penv, penvgen;
        audio = (SoundIn.ar([0]));

        env1 = Env.new([4880, 2035, 7000, 6500, 5484.8],
                        [24, 11, 8.2, 17], 'linear');
        envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

        filter1 =  BLowPass.ar(audio, envgen1);

        env2 = Env.new([719.2, 600, 618, 370, 385, 650, 627.8],
                        [10, 23, 14, 7, 14, 7.2], 'linear');
        envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

        filter2 = BHiPass.ar(filter1, envgen2);

        renv = Env.new([6, 0.56],
                        [60.2], 'linear');
        renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

        reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);
```

```
        sinenv1 = Env.new([1490,  1066, 2386, 1396, 331, 278, 439, 211, 301 ],
                                [ 10.05, 10.05, 10.05,  0.1, 7.55, 7.55, 7.55, 7.55 ], 'sine');
        sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

        sinenv2 = Env.new([496, 403, 577, 310, 492, 331, 206, 301 ],
                                [ 7.55, 7.55, 7.55, 7.55, 0.1, 15.05, 15.05 ], 'sine');
        sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

        ampenv = Env.new([0, 0.4,   0.7,    0.3, 0.6,    0, 0.3,    0.6, 0,   0.2, 0.4, 0.1,  0.7, 0.4, 0.5,   0, 0.6,
                                0.1, 0.3,   0 ],
                            [   3.5, 1.7,  8.3,  6.3, 3.7,  2.8, 7.8, 4.1, 2.4,  0.4, 1.6,   1.3, 1.6, 1.4, 6.7,
                                10.4, 2.5, 1.7, 0.5], 'linear');
        ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

        modulation = (SinOsc.ar([sinenvgen1,sinenvgen2], 0, ampenvgen));
        process = modulation * reverb;

        penv = Env.new([0, 0.4, -0.2, 1, 0, -1, -0.2, 0, 0.2, 0],
                                [ 18.5, 15, 3.5, 7.3, 16.7 ], 'linear');
        penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

        env3 = Env.new([0, 0.728, 0.75488, 0],
                        [0.1, 60, 0.1], 'linear');
        envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

        Out.ar(out, Pan2.ar(process, penvgen, envgen3))
            }).add;



~k = Buffer.alloc(s, 44100 * 3, 1, 0, 20);

SynthDef(\lowrec11, { arg bufnum = 20;
        var audio, rec;
        audio = (SoundIn.ar([0]));
        rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;



SynthDef (\lowplay11a, { arg out = 0, amp=0.3, pan= -0.23, bufnum = 20 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 0.568);
        reverb = FreeVerb.ar(audio, 0.32, 0.32, 0.32);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 4.88, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\lowplay11b, { arg out = 0, amp=0.3, pan= 0.35, bufnum = 20 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 0.458);
        reverb = FreeVerb.ar(audio, 0.5, 0.5, 0.5);
```

```
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 6.01, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;



a = Pbind(*[
        instrument:     "wunderkammer11",
        freq:           Pseq([400], 1),
        dur:            Pseq([61], 1),
]);


c = Prout({

        0.1.wait;
        Synth(\lowrec11).postln;
        SystemClock.sched(0.0,{ arg time; time.postln});
        15.29.wait;
        Synth(\lowplay11a).postln;
        14.71.wait;
        16.57.wait;
        Synth(\lowplay11b).postln;
        20.wait;
        ~k.freeMsg

});

~button11 = Ppar([a, c], 1);

);


// end button 11


//processing activated by button 12
(

SynthDef("wunderkammer12", { arg out=0, amp=0.1, pan=0, freq = 400;
                var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                        reverb,
            sinenv1, sinenvgen1, sinenv2, sinenvgen2,
            modulation, process, ampenv, ampenvgen, penv, penvgen;

        audio = (SoundIn.ar([0]));

        env1 = Env.new([5484.8, 19000, 8000, 9000, 6147.2],
                        [8, 21, 18.2, 13], 'linear');
        envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

        filter1 =  BLowPass.ar(audio, envgen1);
```

```
        env2 = Env.new([627.8, 1773, 800, 1360, 1932, 542.7],
                        [22.3, 2.7, 5.5, 8.2, 21.5], 'linear');
        envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

        filter2 = BHiPass.ar(filter1, envgen2);

        renv = Env.new([0.56, 0.52],
                        [60.2], 'linear');
        renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

        reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

        sinenv1 = Env.new([2056, 1962, 924, 456, 829 ],
                                [ 30.05, 0.1, 15.05, 15.05 ], 'sine');
        sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

        sinenv2 = Env.new([520, 473, 1854, 329, 916, 472, 245, 754, 468 ],
                                [ 7.55, 7.55, 7.55, 7.55, 0.1, 10.05, 10.05, 10.05 ], 'sine');
        sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

        ampenv = Env.new([0, 0.7,  0.5,  0.3, 0.6,  0.4, 0.3,  0.1, 0,  0.5, 0.2, 0.5,  0.1, 0.2, 0.5,  0,
                                0.5,  0.5, 0.8,  0 ],
                            [  18.5, 1.7,  0.3,  3.3, 3.7,  2.8, 1.8, 1.1, 2.4,  0.4, 1.6,  5.3, 1.6, 7.4, 6.7,
                                3.4, 2.5, 1.7, 2.5], 'linear');
        ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

        modulation = (SinOsc.ar([sinenvgen1,sinenvgen2], 0, ampenvgen));
        process = modulation * reverb;

        penv = Env.new([0, 1, -1, 0.2, -0.3, 0, 0.2, -0.3, 0.1, 0],
                            [ 20.5, 13, 3.5, 11.3, 12.7 ], 'linear');
        penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

        env3 = Env.new([0, 0.75488, 0.78432, 0],
                        [0.1, 60, 0.1], 'linear');
        envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

        Out.ar(out, Pan2.ar(process, penvgen, envgen3))
            }).add;

~l = Buffer.alloc(s, 44100 * 10, 1, 0, 23);

SynthDef(\highrec12, { arg bufnum = 23;
        var audio, rec;
        audio = (SoundIn.ar([0]));
        rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;

SynthDef (\highplay12a, { arg out = 0, amp=0.3, pan=0, bufnum = 23 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 1.014);
        reverb = FreeVerb.ar(audio, 0.04, 0.04, 0.04);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 9.46, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
```

```
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\highplay12b, { arg out = 0, amp=0.3, pan= 0.12, bufnum = 23 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 1.148);
        reverb = FreeVerb.ar(audio, 0.2, 0.2, 0.2);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 8.31, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;




a = Pbind(*[
        instrument:     "wunderkammer12",
        freq:           Pseq([400], 1),
        dur:            Pseq([61], 1),
]);


d = Prout({

        0.1.wait;
        Synth(\highrec12).postln;
        SystemClock.sched(0.0,{ arg time; time.postln});
        10.35.wait;
        Synth(\highplay12a).postln;
        19.65.wait;
        13.12.wait;
        Synth(\highplay12b).postln;
        20.wait;
        ~l.freeMsg

});

~button12 = Ppar([a, d], 1);

);


// end button 12


//processing activated by button 13
(

SynthDef("wunderkammer13", { arg out=0, amp=0.1, pan=0, freq = 400;
                var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                        reverb,
            sinenv1, sinenvgen1, sinenv2, sinenvgen2, sinenv3, sinenvgen3, sinenv4, sinenvgen4,
            modulation, process, ampenv, ampenvgen, penv, penvgen;
```

```
audio = (SoundIn.ar([0]));

env1 = Env.new([6147.2, 6300, 2116, 2000, 7000, 6867.2],
                    [10.2, 10, 10, 10, 10, 10], 'linear');
envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

filter1 =  BLowPass.ar(audio, envgen1);

env2 = Env.new([542.7, 5300, 1700, 1900, 6000, 463.9],
                    [10.2, 10, 10, 10, 10, 10], 'linear');
envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

filter2 = BHiPass.ar(filter1, envgen2);

renv = Env.new([0.52, 0.48],
                    [60.2], 'linear');
renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

sinenv1 = Env.new([301, 399, 237, 271, 5020,  3221, 4855 ],
                              [ 10.05, 10.05, 10.05, 0.1, 15.05, 15.05 ], 'sine');
sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

sinenv2 = Env.new([301, 357, 238, 402, 271, 2540, 2513 ],
                              [ 7.55, 7.55, 7.55, 7.55, 0.1, 30.05 ], 'sine');
sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

sinenv3 = Env.new([301, 271, 1270, 695, 1895, 1228 ],
                              [ 30.05, 0.1, 10.05, 10.05, 10.05 ], 'sine');
sinenvgen3 = EnvGen.kr(sinenv3, 1.0, doneAction: 2);

sinenv4 = Env.new([301, 214, 403, 234, 271, 20, 56, 31, 47 ],
                              [ 7.55, 7.55, 7.55, 7.55, 0.1, 10.05, 10.05, 10.05 ], 'linear');
sinenvgen4 = EnvGen.kr(sinenv4, 1.0, doneAction: 2);

ampenv = Env.new([0, 0.3,   0.6,    0.1, 0.6,    0.4, 0.7,    0.1, 0.3,   0.6, 0.8, 0.5,  0.1, 0.2, 0.7,   0,
                                0.7,   0.5, 0.3,   0 ],
                          [   6.5, 1.7,  0.3,  6.3, 3.7,  2.8, 1.8, 0.1, 6.4,  1.4, 0.6,   7.3, 1.6, 1.4, 6.7, 7.4,
                                2.5, 1.7, 8.5], 'linear');
ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

modulation = (SinOsc.ar([sinenvgen1,sinenvgen2, sinenvgen3, sinenvgen4], 0, ampenvgen));
process = modulation * reverb;

penv = Env.new([0, 0.1, -0.4, -0.5, 0.3, -0.4, -0.2, 0.7, -0.5, 0],
                          [ 9.5, 18, 8.5, 7.3, 17.7 ], 'linear');
penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

env3 = Env.new([0, 0.75432, 0.76632, 0],
                    [0.1, 60, 0.1], 'linear');
envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

Out.ar(out, Pan2.ar(process, penvgen, envgen3))
```

```
        }).add;


~m = Buffer.alloc(s, 44100 * 3, 1, 0, 24);


SynthDef(\lowrec13, { arg bufnum = 24;
        var audio, rec;
        audio = (SoundIn.ar([0]));
        rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;


SynthDef (\lowplay13a, { arg out = 0, amp=0.3, pan= -0.32, bufnum = 24 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 0.488);
        reverb = FreeVerb.ar(audio, 0.44, 0.44, 0.44);
        env = Env.new([0, 0.5, 0.5, 0],
                [0.2, 5.74, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\lowplay13b, { arg out = 0, amp=0.3, pan= 0.41, bufnum = 24 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 0.42);
        reverb = FreeVerb.ar(audio, 0.6, 0.6, 0.6);
        env = Env.new([0, 0.5, 0.5, 0],
                [0.2, 6.74, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;


a = Pbind(*[
        instrument:     "wunderkammer13",
        freq:           Pseq([400], 1),
        dur:            Pseq([61], 1),
]);



c = Prout({

        0.1.wait;
        Synth(\lowrec13).postln;
        SystemClock.sched(0.0,{ arg time; time.postln});
        16.14.wait;
        Synth(\lowplay13a).postln;
        13.86.wait;
        17.14.wait;
        Synth(\lowplay13b).postln;
        20.wait;
        ~m.freeMsg
```

```
});

~button13 = Ppar([a, c], 1);

);


// end button 13


//processing activated by button 14
(

SynthDef("wunderkammer14", { arg out=0, amp=0.1, pan=0, freq = 400;
	var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen, reverb,
	sinenv1, sinenvgen1, modulation, process, ampenv, ampenvgen, penv, penvgen;

	audio = (SoundIn.ar([0]));

	env1 = Env.new([6867.2, 3402, 4000, 13089, 10000, 2300, 6700, 7644.8],
			[7.2, 8, 12, 5, 15, 7, 6], 'linear');
	envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

	filter1 =  BLowPass.ar(audio, envgen1);

	env2 = Env.new([463.9, 391.4],
			[60.2], 'linear');
	envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

	filter2 = BHiPass.ar(filter1, envgen2);

	renv = Env.new([0.48, 0.44],
			[60.2], 'linear');
	renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

	reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

	sinenv1 = Env.new([211, 345, 101,  181, 362, 302, 486, 299, 331 ],
				[ 10.05, 10.05, 10.05,  0.1, 7.55, 7.55, 7.55, 7.55 ], 'sine');
	sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

	ampenv = Env.new([0, 0.4,   0.7,    0.3, 0,    0, 0.3,    0.6, 0,   0.2, 0, 0,  0.7, 0.4, 0.5,   0, 0.6,   0,
				0.3,   0 ],
			[   3.5, 1.7,  8.3,  6.3, 3.7,  2.8, 7.8, 4.1, 2.4,  0.4, 1.6,   1.3, 1.6, 1.4, 6.7,
				10.4, 2.5, 1.7, 0.5], 'linear');
	ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

	modulation = (SinOsc.ar(sinenvgen1, 0, ampenvgen));
	process = modulation * reverb;

	env3 = Env.new([0, 0.75632, 0.76088, 0],
			[0.1, 60, 0.1], 'linear');
	envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

	penv = Env.new([0, 0, 0.6, -0.5, 0.3, 0, 0, 0.4, -0.5, 0],
```

```
                                        [ 12.5, 15, 9.5, 8.3, 15.7 ], 'linear');
        penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

        Out.ar(out, Pan2.ar(process, penvgen, envgen3))
                }).add;

~n = Buffer.alloc(s, 44100 * 10, 1, 0, 27);

SynthDef(\highrec14, { arg bufnum = 27;
        var audio, rec;
        audio = (SoundIn.ar([0]));
        rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;


SynthDef (\highplay14a, { arg out = 0, amp=0.3, pan= -0.5, bufnum = 27 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 1.944);
        reverb = FreeVerb.ar(audio, 0.7, 0.7, 0.7);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 4.74, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\highplay14b, { arg out = 0, amp=0.3, pan= -0.41, bufnum = 27 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 1.667);
        reverb = FreeVerb.ar(audio, 0.58, 0.58, 0.58);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 5.59, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;



a = Pbind(*[
        instrument:         "wunderkammer14",
        freq:               Pseq([400], 1),
        dur:        Pseq([61], 1),
]);

d = Prout({

        0.1.wait;
        Synth(\highrec14).postln;
        SystemClock.sched(0.0,{ arg time; time.postln});
        21.8.wait;
        Synth(\highplay14a).postln;
        8.2.wait;
        19.71.wait;
        Synth(\highplay14b).postln;
        20.wait;
```

```
        ~n.freeMsg

});

~button14 = Ppar([a, d], 1);

);


// end button 14


//processing activated by button 15
(

SynthDef("wunderkammer15", { arg out=0, amp=0.1, pan=0, freq = 400;
                var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                        reverb,
            sinenv1, sinenvgen1, sinenv2, sinenvgen2, sinenv3, sinenvgen3, sinenv4, sinenvgen4,
            modulation, process, ampenv, ampenvgen, penv, penvgen;

        audio = (SoundIn.ar([0]));

        env1 = Env.new([7644.8, 8480],
                        [60.2], 'linear');
        envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2 );

        filter1 =  BLowPass.ar(audio, envgen1);

        env2 = Env.new([391.4, 617, 480, 500, 1000, 1285, 350, 325.2],
                        [8, 9.2, 12, 20, 13, 4, 2], 'linear');
        envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

        filter2 = BHiPass.ar(filter1, envgen2);

        renv = Env.new([0.44, 0.4],
                        [60.2], 'linear');
        renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

        reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

        sinenv1 = Env.new([3209, 2035, 6894, 2318, 3044, 30, 0 ],
                                [ 7.55, 7.55, 7.55, 7.55, 0.1, 30.05 ], 'sine');
        sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

        sinenv2 = Env.new([2244, 2217, 30, 46, 22, 39, 0 ],
                                [ 30.05, 0.1, 7.55, 7.55, 7.55, 7.55 ], 'sine');
        sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

        sinenv3 = Env.new([812, 687, 986, 770, 30, 46, 21, 0 ],
                                [ 10.05, 10.05, 10.05, 0.1, 10.05, 10.05, 10.05], 'sine');
        sinenvgen3 = EnvGen.kr(sinenv3, 1.0, doneAction: 2);

        sinenv4 = Env.new([326, 264, 353, 30, 57, 0 ],
                                [ 15.05, 15.05,  0.1, 15.05, 15.05 ], 'sine');
```

```
        sinenvgen4 = EnvGen.kr(sinenv4, 1.0, doneAction: 2);

        ampenv = Env.new([0, 0.3,  0.3,  0.5, 0.5,  0.2, 0.7,  0.3, 0.2,  0.6, 0.3, 0.1,  0.6, 0.4, 0.7,  0.2,
                                0.1,  0.3, 0.6,  0 ],
                            [  10.5, 1.7,  0.3,  6.3, 3.7,  2.8, 1.8, 4.1, 2.4,  0.4, 1.6,  7.3, 1.6, 1.4, 6.7,
                                3.4, 2.5, 1.7, 8.5], 'linear');
        ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

        modulation = (SinOsc.ar([sinenvgen1,sinenvgen2, sinenvgen3, sinenvgen4], 0, ampenvgen));
        process = modulation * reverb;

        env3 = Env.new([0, 0.7535088, 0.76388, 0],
                        [0.1, 60, 0.1], 'linear');
        envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

        penv = Env.new([0, -0.1, -0.3, 0.3, -0.3, -0.4, -0.2, 0.9, 0.5, 0],
                            [ 18.5, 25, 9.5, 2.3, 5.7 ], 'linear');
        penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

        Out.ar(out, Pan2.ar(process, penvgen, envgen3))
            }).add;

~o = Buffer.alloc(s, 44100 * 3, 1, 0, 28);

SynthDef(\lowrec15, { arg bufnum = 28;
        var audio, rec;
        audio = (SoundIn.ar([0]));
        rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;



SynthDef (\lowplay15a, { arg out = 0, amp=0.3, pan=0.26, bufnum = 28 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 0.525);
        reverb = FreeVerb.ar(audio, 0.38, 0.38, 0.38);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 5.31, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\lowplay15b, { arg out = 0, amp=0.3, pan= 0.03, bufnum = 28 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 0.875);
        reverb = FreeVerb.ar(audio, 0.66, 0.66, 0.66);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 3.02, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;



a = Pbind(*[
```

```
        instrument:     "wunderkammer15",
        freq:           Pseq([400], 1),
        dur:            Pseq([61], 1),
]);


c = Prout({

        0.1.wait;
        Synth(\lowrec15).postln;
        SystemClock.sched(0.0,{ arg time; time.postln});
        15.71.wait;
        Synth(\lowplay15a).postln;
        14.29.wait;
        13.43.wait;
        Synth(\lowplay15b).postln;
        20.wait;
        ~o.freeMsg

});

~button15 = Ppar([a, c], 1);

);


// end button 15


//processing activated by button 16
(

SynthDef("wunderkammer16", { arg out=0, amp=0.1, pan=0, freq = 400;
                var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                        reverb,
            sinenv1, sinenvgen1, sinenv2, sinenvgen2,
            modulation, process, ampenv, ampenvgen, penv, penvgen;
        audio = (SoundIn.ar([0]));

        env1 = Env.new([8480, 7000, 7050, 5281, 3119, 11000, 9372.8],
                        [11, 27.2, 2, 6, 1, 13], 'linear');
        envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

        filter1 =  BLowPass.ar(audio, envgen1);

        env2 = Env.new([325.2, 5920, 6172, 517, 680, 112, 200, 265.3],
                        [6.2, 17, 13, 5, 4, 11, 4], 'linear');
        envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

        filter2 = BHiPass.ar(filter1, envgen2);

        renv = Env.new([0.4, 0.36],
                        [60.2], 'linear');
        renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);
```

```
        reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

        sinenv1 = Env.new([1867, 3869, 1773, 641, 322, 546 ],
                                [ 15.05, 15.05,  0.1, 15.05, 15.05 ], 'sine');
        sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

        sinenv2 = Env.new([512, 354, 967, 508, 460, 303, 567, 456 ],
                                [ 10.05, 10.05, 10.05, 0.1, 310.05, 10.05, 10.05 ], 'sine');
        sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

        ampenv = Env.new([0, 0.3,   0.6,    0.4, 0.5,    0.2, 0.7,    0.3, 0.4,   0.3, 0.5, 0.2,  0.1, 0.4, 0.6,   0.2,
                                0.1,   0.5, 0.3,   0 ],
                            [   16.5, 1.7,  6.3,  6.3, 3.7,  2.8, 1.8, 4.1, 2.4,  0.4, 1.6,   12.3, 1.6, 1.4, 6.7,
                                3.4, 2.5, 1.7, 3.5], 'linear');
        ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

        modulation = (SinOsc.ar([sinenvgen1,sinenvgen2], 0, ampenvgen));
        process = modulation * reverb;

        penv = Env.new([0, -0.5, -0.4, 0.5, 0.3, -0.6, 0.2, 0.1, 0.6, 0],
                            [ 5.5, 15, 16.5, 9.3, 14.7 ], 'linear');
        penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

        env3 = Env.new([0, 0.76388, 0.7742768, 0],
                        [0.1, 60, 0.1], 'linear');
        envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

        Out.ar(out, Pan2.ar(process, penvgen, envgen3))
            }).add;


~p = Buffer.alloc(s, 44100 * 10, 1, 0, 31);


SynthDef(\highrec16, { arg bufnum = 31;
        var audio, rec;
        audio = (SoundIn.ar([0]));
        rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;


SynthDef (\highplay16a, { arg out = 0, amp=0.3, pan= -0.7, bufnum = 31 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 3.182);
        reverb = FreeVerb.ar(audio, 0.98, 0.98, 0.98);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 2.74, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\highplay16b, { arg out = 0, amp=0.3, pan= -0.32, bufnum = 31 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 1.458);
        reverb = FreeVerb.ar(audio, 0.46, 0.46, 0.46);
```

```
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 6.45, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;




a = Pbind(*[
        instrument:     "wunderkammer16",
        freq:           Pseq([400], 1),
        dur:            Pseq([61], 1),
]);

d = Prout({

        0.1.wait;
        Synth(\highrec16).postln;
        SystemClock.sched(0.0,{ arg time; time.postln});
        26.65.wait;
        Synth(\highplay16a).postln;
        3.35.wait;
        17.63.wait;
        Synth(\highplay16b).postln;
        20.wait;
        ~p.freeMsg

});

~button16 = Ppar([a, d], 1);

);


// end button 16


//processing activated by button 17
(

SynthDef("wunderkammer17", { arg out=0, amp=0.1, pan=0, freq = 400;
                var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                        reverb,
            sinenv1, sinenvgen1, sinenv2, sinenvgen2, sinenv3, sinenvgen3, sinenv4, sinenvgen4,
            modulation, process, ampenv, ampenvgen, penv, penvgen;

        audio = (SoundIn.ar([0]));

        env1 = Env.new([9372.8, 10323.2],
                        [60.2], 'linear');
        envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

        filter1 =  BLowPass.ar(audio, envgen1);
```

```
        env2 = Env.new([265.3, 211.7],
                        [60.2], 'linear');
        envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

        filter2 = BHiPass.ar(filter1, envgen2);

        renv = Env.new([0.36, 0.32],
                        [60.2], 'linear');
        renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

        reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

        sinenv1 = Env.new([4855, 3206, 4691, 735, 877, 439, 641 ],
                                [ 15.05, 15.05,  0.1, 10.05, 10.05, 10.05 ], 'sine');
        sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

        sinenv2 = Env.new([2513, 3895, 1133, 2486, 735, 641 ],
                                [ 10.05, 10.05, 10.05,  0.1, 30.05 ], 'sine');
        sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

        sinenv3 = Env.new([1228,  1784, 950, 1855, 1187, 464,  679, 460 ],
                                [ 7.55, 7.55, 7.55, 7.55,0.1, 15.05, 15.05 ], 'sine');
        sinenvgen3 = EnvGen.kr(sinenv3, 1.0, doneAction: 2);

        sinenv4 = Env.new([47, 76, 464,  366, 509, 285, 460 ],
                                [ 30.05, 0.1, 7.55, 7.55, 7.55, 7.55 ], 'sine');
        sinenvgen4 = EnvGen.kr(sinenv4, 1.0, doneAction: 2);

        ampenv = Env.new([0, 0.7,   0.3,    0, 0.7,    0.2, 0.2,    0.1, 0.3,   0.7, 0.4, 0.1,  0.2, 0.8, 0.6,   0,
                                0.2,   0.5, 0.3,   0 ],
                            [   1.5, 1.7,  1.3,  5.3, 4.7,  0.8, 2.8, 12.1, 1.4,  4.4, 0.6,   9.3, 3.6, 1.4, 7.7,
                                3.4, 3.5, 1.7, 1.5], 'linear');
        ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

        modulation = (SinOsc.ar([sinenvgen1,sinenvgen2, sinenvgen3, sinenvgen4], 0, ampenvgen));
        process = modulation * reverb;

        env3 = Env.new([0, 0.7742768, 0.7846942, 0],
                        [0.1, 60, 0.1], 'linear');
        envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

        penv = Env.new([0, 0.3, 0.4, -0.5, 0.3, 0.5, 0.2, -0.7, -0.9, 0],
                                [ 12.5, 21, 3.5, 5.3, 18.7 ], 'linear');
        penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

        Out.ar(out, Pan2.ar(process, penvgen, envgen3))
                }).add;



~q = Buffer.alloc(s, 44100 * 3, 1, 0, 32);

SynthDef(\lowrec17, { arg bufnum = 32;
        var audio, rec;
```

```
        audio = (SoundIn.ar([0]));
        rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;

SynthDef (\lowplay17a, { arg out = 0, amp=0.3, pan=0.29, bufnum = 32 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 0.5);
        reverb = FreeVerb.ar(audio, 0.42, 0.42, 0.42);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 5.6, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\lowplay17b, { arg out = 0, amp=0.3, pan= 0.58, bufnum = 32 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 0.339);
        reverb = FreeVerb.ar(audio, 0.84, 0.84, 0.84);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 8.44, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;


a = Pbind(*[
        instrument:     "wunderkammer17",
        freq:           Pseq([400], 1),
        dur:            Pseq([61], 1),
]);


c = Prout({

        0.1.wait;
        Synth(\lowrec17).postln;
        SystemClock.sched(0.0,{ arg time; time.postln});
        16.wait;
        Synth(\lowplay17a).postln;
        14.wait;
        18.86.wait;
        Synth(\lowplay17b).postln;
        20.wait;
        ~q.freeMsg

});


~button17 = Ppar([a, c], 1);

);


// end button 17
```

```
//processing activated by button 18
(

SynthDef("wunderkammer18", { arg out=0, amp=0.1, pan=0, freq = 400;
                var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                        reverb,
            sinenv1, sinenvgen1, sinenv2, sinenvgen2, sinenv3, sinenvgen3, sinenv4, sinenvgen4,
            modulation, process, ampenv, ampenvgen, penv, penvgen;

        audio = (SoundIn.ar([0]));

        env1 = Env.new([10323.2, 8300, 5193, 7000, 7060, 2115, 2200, 8000, 11331.2],
                        [8.2, 9, 9, 13, 6, 3, 7, 5], 'linear');
        envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

        filter1 =  BLowPass.ar(audio, envgen1);

        env2 = Env.new([211.7, 7000, 2000, 380, 6000, 513, 1700, 5000, 164.4],
                        [8.2, 9, 9, 13, 6, 3, 7, 5], 'linear');
        envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

        filter2 = BHiPass.ar(filter1, envgen2);

        renv = Env.new([0.32, 0.28],
                        [60.2], 'linear');
        renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

        reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

        sinenv1 = Env.new([4032, 2856, 5889, 2309, 3867, 1962,  1288, 1867 ],
                                [ 7.55, 7.55, 7.55, 7.55, 0.1, 15.05, 15.05 ], 'sine');
        sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

        sinenv2 = Env.new([2379, 3497, 1765, 2352, 1962, 899, 1728, 1197, 1867 ],
                                [ 10.05, 10.05, 10.05, 0.1, 7.55, 7.55, 7.55, 7.55 ], 'sine');
        sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

        sinenv3 = Env.new([1020, 978, 516, 613, 407, 512 ],
                                [ 30.05, 0.1, 10.05, 10.05, 10.05 ], 'sine');
        sinenvgen3 = EnvGen.kr(sinenv3, 1.0, doneAction: 2);

        sinenv4 = Env.new([187, 97, 214, 516, 512 ],
                                [ 15.05, 15.05, 0.1, 30.05 ], 'sine');
        sinenvgen4 = EnvGen.kr(sinenv4, 1.0, doneAction: 2);

        ampenv = Env.new([0, 0.3,   0.3,    0, 0.3,    0.7, 0.2,    0.7, 0.2,   0.6, 0.5, 0.1,  0.2, 0.7, 0.2,   0,
                                0.6,   0.2, 0.6,   0 ],
                        [   6.5, 1.7,  0.3,  6.3, 3.7,  2.8, 1.8, 0.1, 6.4,  1.4, 0.6,   7.3, 1.6, 1.4, 6.7, 7.4,
                                2.5, 1.7, 8.5], 'linear');
        ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

        modulation = (SinOsc.ar([sinenvgen1,sinenvgen2, sinenvgen3, sinenvgen4], 0, ampenvgen));
        process = modulation * reverb;
```

```
		env3 = Env.new([0, 0.8146992, 0.8253808, 0],
				[0.1, 60, 0.1], 'linear');
		envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

		penv = Env.new([0, 1, 0.4, -0.5, -0.3, -1, -0.2, 0.8, 0.5, 0],
					[ 7.5, 20, 7.5, 8.3, 17.7 ], 'linear');
		penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

		Out.ar(out, Pan2.ar(process, penvgen, envgen3))
			}).add;


~r = Buffer.alloc(s, 44100 * 10, 1, 0, 35);



SynthDef(\highrec18, { arg bufnum = 35;
		var audio, rec;
		audio = (SoundIn.ar([0]));
		rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;


SynthDef (\highplay18a, { arg out = 0, amp=0.3, pan= -0.26, bufnum = 35 ;
		var audio, reverb, env, envgen;
		audio = PlayBuf.ar(1, bufnum, 1.346);
		reverb = FreeVerb.ar(audio, 0.38, 0.38, 0.38);
		env = Env.new([0, 0.5, 0.5, 0],
				[0.2, 7.02, 0.2 ], 'linear');
		envgen = EnvGen.kr(env, 1.0, doneAction: 2);
		Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\highplay18b, { arg out = 0, amp=0.3, pan= -0.15, bufnum = 35 ;
		var audio, reverb, env, envgen;
		audio = PlayBuf.ar(1, bufnum, 1.167);
		reverb = FreeVerb.ar(audio, 0.22, 0.22, 0.22);
		env = Env.new([0, 0.5, 0.5, 0],
				[0.2, 8.16, 0.2 ], 'linear');
		envgen = EnvGen.kr(env, 1.0, doneAction: 2);
		Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;




a = Pbind(*[
		instrument:		"wunderkammer18",
		freq:			Pseq([400], 1),
		dur:			Pseq([61], 1),
]);


d = Prout({
```

```
            0.1.wait;
            Synth(\highrec18).postln;
            SystemClock.sched(0.0,{ arg time; time.postln});
            16.24.wait;
            Synth(\highplay18a).postln;
            13.76.wait;
            13.47.wait;
            Synth(\highplay18b).postln;
            20.wait;
            ~r.freeMsg

});

~button18 = Ppar([a, d], 1);

);


// end button 18


//processing activated by button 19
(

SynthDef("wunderkammer19", { arg out=0, amp=0.1, pan=0, freq = 400;
                        var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                                reverb,
                    sinenv1, sinenvgen1, sinenv2, sinenvgen2, sinenv3, sinenvgen3, sinenv4, sinenvgen4,
                    modulation, process, ampenv, ampenvgen, penv, penvgen;

            audio = (SoundIn.ar([0]));

            env1 = Env.new([11331.2, 1200, 750, 600, 12951.2],
                                [10.2, 20, 20, 10, ], 'linear');
            envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

            filter1 =  BLowPass.ar(audio, envgen1);

            env2 = Env.new([164.4, 123.5],
                                [60.2], 'linear');
            envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

            filter2 = BHiPass.ar(filter1, envgen2);

            renv = Env.new([0.28, 0.24],
                                [60.2], 'linear');
            renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

            reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

            sinenv1 = Env.new([121, 45, 286,  90, 3044, 2311, 4503, 2879 ],
                                        [ 10.05, 10.05, 10.05, 0.1, 10.05, 10.05, 10.05 ], 'sine');
            sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);
```

```
        sinenv2 = Env.new([121, 199, 45, 187, 90, 2217, 1432, 2190 ],
                                [ 7.55, 7.55, 7.55, 7.55, 0.1, 15.05, 15.05 ], 'sine');
        sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

        sinenv3 = Env.new([121, 64, 90, 770, 365, 907, 412, 728 ],
                                [ 15.05, 15.05, 0.1, 7.55, 7.55, 7.55, 7.55 ], 'sine');
        sinenvgen3 = EnvGen.kr(sinenv3, 1.0, doneAction: 2);

        sinenv4 = Env.new([121, 90, 353, 381 ],
                                [ 30.05, 0.1, 30.05 ], 'sine');
        sinenvgen4 = EnvGen.kr(sinenv4, 1.0, doneAction: 2);

        ampenv = Env.new([0, 0.3,   0.6,    0.3, 0.6,    0, 0.3,    0.7, 0,   0.3, 0, 0.5,  0, 0.4, 0.7,   0, 0.5,   0,
                                0.7,   0 ],
                            [   17.5, 1.7,  0.3,  4.3, 2.7,  1.8, 1.8, 3.1, 1.4,  0.4, 1.6,   7.3, 1.6, 1.4, 6.7,
                                2.4, 2.5, 1.7, 8.5], 'linear');
        ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

        modulation = (SinOsc.ar([sinenvgen1,sinenvgen2, sinenvgen3, sinenvgen4], 0, ampenvgen));
        process = modulation * reverb;

        env3 = Env.new([0, 0.8254172, 0.8356208, 0],
                        [0.1, 60, 0.1], 'linear');
        envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

        penv = Env.new([0, -0.1, 0.7, -0.5, 0.3, 0, -0.2, 0.3, 1, 0],
                            [ 17.5, 15, 3.5, 14.3, 10.7 ], 'linear');
        penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

        Out.ar(out, Pan2.ar(process, penvgen, envgen3))
            }).add;



~s = Buffer.alloc(s, 44100 * 3, 1, 0, 36);

SynthDef(\lowrec19, { arg bufnum = 36;
        var audio, rec;
        audio = (SoundIn.ar([0]));
        rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;


SynthDef (\lowplay19a, { arg out = 0, amp=0.3, pan= -0.2, bufnum = 36 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 0.6);
        reverb = FreeVerb.ar(audio, 0.28, 0.28, 0.28);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 4.6, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\lowplay19b, { arg out = 0, amp=0.3, pan= 0.12, bufnum = 36 ;
```

```
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 0.7);
        reverb = FreeVerb.ar(audio, 0.18, 0.18, 0.18);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 3.88, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;



a = Pbind(*[
        instrument:         "wunderkammer19",
        freq:               Pseq([400], 1),
        dur:        Pseq([61], 1),
]);


c = Prout({

        0.1.wait;
        Synth(\lowrec19).postln;
        SystemClock.sched(0.0,{ arg time; time.postln});
        15.wait;
        Synth(\lowplay19a).postln;
        15.wait;
        14.29.wait;
        Synth(\lowplay19b).postln;
        20.wait;
        ~s.freeMsg

});


~button19 = Ppar([a, c], 1);

);


// end button 19


//processing activated by button 20
(

SynthDef("wunderkammer20", { arg out=0, amp=0.1, pan=0, freq = 400;
                var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                                reverb,
            sinenv1, sinenvgen1, sinenv2, sinenvgen2,
            modulation, process, ampenv, ampenvgen, penv, penvgen;

        audio = (SoundIn.ar([0]));

        env1 = Env.new([12951.2, 14103.2],
                        [60.2], 'linear');
```

```
        envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

        filter1 =  BLowPass.ar(audio, envgen1);

        env2 = Env.new([123.5, 8003, 11574, 10140, 88.8],
                        [10.2, 23, 10, 17], 'linear');
        envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

        filter2 = BHiPass.ar(filter1, envgen2);

        renv = Env.new([0.24, 0.2],
                        [60.2], 'linear');
        renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

        reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

        sinenv1 = Env.new([1584, 1134, 1986, 954, 1490, 392, 105, 362 ],
                                [ 7.55, 7.55, 7.55, 7.55,  0.1, 15.05, 15.05 ], 'sine');
        sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

        sinenv2 = Env.new([500, 764, 321, 496, 392, 112, 765, 362 ],
                                [ 10.05, 10.05, 10.05, 0.1, 10.05, 10.05, 10.05 ], 'sine');
        sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

        ampenv = Env.new([0, 0.2,   0.3,    0.7, 0.3,    0.1, 0.6,    0, 0.4,   0.2, 0.8, 0.1,  0.1, 0.4, 0.7,   0.3,
                                0.2,   0.2, 0.6,   0 ],
                            [   5.5, 1.7,  0.3,  1.3, 3.7,  4.8, 1.8, 6.1, 6.4,  2.4, 1.6,   12.3, 1.6, 1.4, 6.7,
                                3.4, 2.5, 1.7, 4.5], 'linear');
        ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

        modulation = (SinOsc.ar([sinenvgen1,sinenvgen2], 0, ampenvgen));
        process = modulation * reverb;

        env3 = Env.new([0, 0.8254172, 0.8356208, 0],
                        [0.1, 60, 0.1], 'linear');
        envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

        penv = Env.new([0, 0.3, -0.2, 0.1, -0.3, 0.4, -0.2, 0.1, -0.5, 0],
                            [ 11.5, 16, 4.5, 13.3, 15.7 ], 'linear');
        penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

        Out.ar(out, Pan2.ar(process, penvgen, envgen3))
            }).add;

~t = Buffer.alloc(s, 44100 * 10, 1, 0, 39);


SynthDef(\highrec20, { arg bufnum = 39;
        var audio, rec;
        audio = (SoundIn.ar([0]));
        rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;


SynthDef (\highplay20a, { arg out = 0, amp=0.3, pan= -0.09, bufnum = 39 ;
```

```
	var audio, reverb, env, envgen;
	audio = PlayBuf.ar(1, bufnum, 1.094);
	reverb = FreeVerb.ar(audio, 0.14, 0.14, 0.14);
	env = Env.new([0, 0.5, 0.5, 0],
			[0.2, 8.74, 0.2 ], 'linear');
	envgen = EnvGen.kr(env, 1.0, doneAction: 2);
	Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\highplay20b, { arg out = 0, amp=0.3, pan= 0.47, bufnum = 39 ;
	var audio, reverb, env, envgen;
	audio = PlayBuf.ar(1, bufnum, 1.872);
	reverb = FreeVerb.ar(audio, 0.68, 0.68, 0.68);
	env = Env.new([0, 0.5, 0.5, 0],
			[0.2, 4.94, 0.2 ], 'linear');
	envgen = EnvGen.kr(env, 1.0, doneAction: 2);
	Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;




a = Pbind(*[
	instrument:	"wunderkammer20",
	freq:		Pseq([400], 1),
	dur:	Pseq([61], 1),
]);


d = Prout({

	0.1.wait;
	Synth(\highrec20).postln;
	SystemClock.sched(0.0,{ arg time; time.postln});
	12.08.wait;
	Synth(\highplay20a).postln;
	17.92.wait;
	21.45.wait;
	Synth(\highplay20b).postln;
	20.wait;
	~t.freeMsg

});

~button20 = Ppar([a, d], 1);

);


// end button 20


//processing activated by button 21
```

```
(

SynthDef("wunderkammer21", { arg out=0, amp=0.1, pan=0, freq = 400;
                    var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                                reverb,
                        sinenv1, sinenvgen1, sinenv2, sinenvgen2, sinenv3, sinenvgen3, sinenv4, sinenvgen4,
                        modulation, process, ampenv, ampenvgen, penv, penvgen;

                audio = (SoundIn.ar([0]));

                env1 = Env.new([14103.2, 12000, 600, 4599, 17000, 15312.8],
                                [10.2, 7, 15, 13, 5, 10], 'linear');
                envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

                filter1 =  BLowPass.ar(audio, envgen1);

                env2 = Env.new([88.8, 513, 170, 60, 575, 60.4],
                                [5.2, 20, 5, 5, 2, 23], 'linear');
                envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

                filter2 = BHiPass.ar(filter1, envgen2);

                renv = Env.new([0.2, 0.16],
                                [60.2], 'linear');
                renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

                reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

                sinenv1 = Env.new([1018, 465, 1687, 924, 4361, 2316, 6503, 4197 ],
                                        [ 10.05, 10.05, 10.05, 0.1, 10.05, 10.05, 10.05 ], 'sine');
                sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

                sinenv2 = Env.new([1018, 659, 1754, 436, 924, 2432, 2406 ],
                                        [ 7.55, 7.55, 7.55, 7.55, 0.1, 30.05 ], 'sine');
                sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

                sinenv3 = Env.new([476,  689, 472, 1103, 454, 1062 ],
                                        [ 15.05, 15.05,  0.1, 15.05, 15.05], 'sine');
                sinenvgen3 = EnvGen.kr(sinenv3, 1.0, doneAction: 2);

                sinenv4 = Env.new([476, 472, 131, 76, 203, 22, 139 ],
                                        [ 30.05, 0.1, 7.55, 7.55, 7.55, 7.55 ], 'sine');
                sinenvgen4 = EnvGen.kr(sinenv4, 1.0, doneAction: 2);

                ampenv = Env.new([0, 0.3,   0.2,    0, 0.6,    0.2, 0,    0.4, 0.6,   0.3, 0.2, 0.5,  0.2, 0.7, 0.3,   0.4,
                                        0.1,   0.5, 0.3,   0 ],
                                [   1.5, 1.7,  2.3,  8.3, 3.7,  4.8, 1.8, 4.1, 2.4,  1.4, 3.6,   1.3, 1.6, 1.4, 6.7, 5.4,
                                        2.5, 1.7, 12.5], 'linear');
                ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

                modulation = (SinOsc.ar([sinenvgen1,sinenvgen2, sinenvgen3, sinenvgen4], 0, ampenvgen));
                process = modulation * reverb;

                env3 = Env.new([0, 0.8254172, 0.8356208, 0],
                                [0.1, 60, 0.1], 'linear');
```

```
		envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

		penv = Env.new([0, 0.1, -0.3, -0.2, 0.9, 0.4, -0.4, -0.8, -0.5, 0],
					[ 4.5, 19, 5.5, 16.3, 15.7 ], 'linear');
		penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

		Out.ar(out, Pan2.ar(process, penvgen, envgen3))
			}).add;

~u = Buffer.alloc(s, 44100 * 3, 1, 0, 40);

SynthDef(\lowrec21, { arg bufnum = 40;
		var audio, rec;
		audio = (SoundIn.ar([0]));
		rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;

SynthDef (\lowplay21a, { arg out = 0, amp=0.3, pan=0.67, bufnum = 40 ;
		var audio, reverb, env, envgen;
		audio = PlayBuf.ar(1, bufnum, 0.309);
		reverb = FreeVerb.ar(audio, 0.96, 0.96, 0.96);
		env = Env.new([0, 0.5, 0.5, 0],
				[0.2, 9.3, 0.2 ], 'linear');
		envgen = EnvGen.kr(env, 1.0, doneAction: 2);
		Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\lowplay21b, { arg out = 0, amp=0.3, pan= -0.15, bufnum = 40 ;
		var audio, reverb, env, envgen;
		audio = PlayBuf.ar(1, bufnum, 0.677);
		reverb = FreeVerb.ar(audio, 0.2, 0.2, 0.2);
		env = Env.new([0, 0.5, 0.5, 0],
				[0.2, 4.03, 0.2 ], 'linear');
		envgen = EnvGen.kr(env, 1.0, doneAction: 2);
		Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;




a = Pbind(*[
		instrument:		"wunderkammer21",
		freq:			Pseq([400], 1),
		dur:			Pseq([61], 1),
]);



c = Prout({

		0.1.wait;
		Synth(\lowrec21).postln;
```

```
        SystemClock.sched(0.0,{ arg time; time.postln});
        19.71.wait;
        Synth(\lowplay21a).postln;
        10.29.wait;
        14.43.wait;
        Synth(\lowplay21b).postln;
        20.wait;
        ~u.freeMsg

});

~button21 = Ppar([a, c], 1);

);


// end button 21


//processing activated by button 22
(

SynthDef("wunderkammer22", { arg out=0, amp=0.1, pan=0, freq = 400;
                var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                        reverb,
            sinenv1, sinenvgen1, sinenv2, sinenvgen2, sinenv3, sinenvgen3, sinenv4, sinenvgen4,
            modulation, process, ampenv, ampenvgen, penv, penvgen;

        audio = (SoundIn.ar([0]));

        env1 = Env.new([15312.8, 16580],
                        [60.2], 'linear');
        envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

        filter1 =  BLowPass.ar(audio, envgen1);

        env2 = Env.new([60.4, 38.4],
                        [60.2], 'linear');
        envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

        filter2 = BHiPass.ar(filter1, envgen2);

        renv = Env.new([0.16, 0.12],
                        [60.2], 'linear');
        renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

        reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

        sinenv1 = Env.new([90, 23, 183, 42,  60, 3373, 3209 ],
                                [ 7.55, 7.55, 7.55, 7.55,  0.1, 30.05 ], 'sine');
        sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

        sinenv2 = Env.new([90, 60, 2271, 1034, 2854, 970,  2244 ],
                                [ 30.05, 0.1, 7.55, 7.55, 7.55, 7.55], 'sine');
        sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);
```

```
		sinenv3 = Env.new([90,  178, 43, 60, 853, 523, 1654, 812 ],
					[ 10.05, 10.05, 10.05, 0.1, 10.05, 10.05, 10.05 ], 'sine');
		sinenvgen3 = EnvGen.kr(sinenv3, 1.0, doneAction: 2);

		sinenv4 = Env.new([90, 21, 60, 298, 143, 326 ],
					[ 15.05, 15.05, 0.1, 15.05, 15.05 ], 'sine');
		sinenvgen4 = EnvGen.kr(sinenv4, 1.0, doneAction: 2);

		ampenv = Env.new([0, 0.7,  0.6,  0.3, 0.2,  0.4, 0.3,  0.1, 0.2,  0.6, 0.8, 0.3,  0.4, 0.4, 0.2,  0.1,
					0.6,  0.5, 0.3,  0 ],
				[  10.5, 1.7,  0.3,  6.3, 3.7,  2.8, 1.8, 4.1, 2.4,  0.4, 1.6,  7.3, 1.6, 1.4, 6.7,
					3.4, 2.5, 1.7, 8.5], 'linear');
		ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

		modulation = (SinOsc.ar([sinenvgen1,sinenvgen2, sinenvgen3, sinenvgen4], 0, ampenvgen));
		process = modulation * reverb;

		env3 = Env.new([0, 0.8254172, 0.8356208, 0],
				[0.1, 60, 0.1], 'linear');
		envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

		penv = Env.new([0, 0.1, 0.7, -0.5, -0.3, 0.7, 0.2, 0.3, -0.4, 0],
					[ 2.5, 5, 16.5, 15.3, 21.7 ], 'linear');
		penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

		Out.ar(out, Pan2.ar(process, penvgen, envgen3))
		}).add;


~v = Buffer.alloc(s, 44100 * 10, 1, 0, 43);


SynthDef(\highrec22, { arg bufnum = 43;
		var audio, rec;
		audio = (SoundIn.ar([0]));
		rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;

SynthDef (\highplay22a, { arg out = 0, amp=0.3, pan=0.64, bufnum = 43 ;
		var audio, reverb, env, envgen;
		audio = PlayBuf.ar(1, bufnum, 2.8);
		reverb = FreeVerb.ar(audio, 0.92, 0.92, 0.92);
		env = Env.new([0, 0.5, 0.5, 0],
				[0.2, 3.17, 0.2 ], 'linear');
		envgen = EnvGen.kr(env, 1.0, doneAction: 2);
		Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\highplay22b, { arg out = 0, amp=0.3, pan= -0.23, bufnum = 43 ;
		var audio, reverb, env, envgen;
		audio = PlayBuf.ar(1, bufnum, 1.296);
		reverb = FreeVerb.ar(audio, 0.34, 0.34, 0.34);
		env = Env.new([0, 0.5, 0.5, 0],
				[0.2, 7.31, 0.2 ], 'linear');
```

```
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;




a = Pbind(*[
        instrument:         "wunderkammer22",
        freq:               Pseq([400], 1),
        dur:        Pseq([61], 1),
]);

d = Prout({

        0.1.wait;
        Synth(\highrec22).postln;
        SystemClock.sched(0.0,{ arg time; time.postln});
        25.61.wait;
        Synth(\highplay22a).postln;
        4.39.wait;
        15.55.wait;
        Synth(\highplay22b).postln;
        20.wait;
        ~v.freeMsg

});

~button22 = Ppar([a, d], 1);

);


// end button 22


//processing activated by button 23
(

SynthDef("wunderkammer23", { arg out=0, amp=0.1, pan=0, freq = 400;
                var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                        reverb,
            sinenv1, sinenvgen1, sinenv2, sinenvgen2, sinenv3, sinenvgen3, sinenv4, sinenvgen4,
            modulation, process, ampenv, ampenvgen, penv, penvgen;

        audio = (SoundIn.ar([0]));

        env1 = Env.new([16580, 18000, 8094, 19809, 10000, 12300, 10700, 17904.8],
                        [7.2, 10, 10, 15, 5, 10, 3], 'linear');
        envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

        filter1 =  BLowPass.ar(audio, envgen1);

        env2 = Env.new([38.4, 123, 480, 42, 55, 319, 350, 22.6],
```

```
                        [8, 9.2, 22, 10, 3, 4, 12], 'linear');
        envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

        filter2 = BHiPass.ar(filter1, envgen2);

        renv = Env.new([0.12, 0.08],
                        [60.2], 'linear');
        renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

        reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

        sinenv1 = Env.new([3538, 1543, 3373, 1396, 867, 1301 ],
                                [ 15.05, 15.05, 0.1, 15.05, 15.05 ], 'sine');
        sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

        sinenv2 = Env.new([2298, 2271, 1396, 1301 ],
                                [ 30.05, 0.1, 30.05 ], 'sine');
        sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

        sinenv3 = Env.new([895, 499, 1029, 853, 492, 698, 266, 488 ],
                                [ 10.05, 10.05, 10.05, 0.1, 10.05, 10.05, 10.05 ], 'sine');
        sinenvgen3 = EnvGen.kr(sinenv3, 1.0, doneAction: 2);

        sinenv4 = Env.new([270, 145, 499, 127, 298, 492, 328, 783, 400, 488 ],
                                [7.55, 7.55, 7.55, 7.55, 0.1, 7.55, 7.55, 7.55, 7.55, ], 'sine');
        sinenvgen4 = EnvGen.kr(sinenv4, 1.0, doneAction: 2);

        ampenv = Env.new([0, 0.7,   0.3,    0, 0.7,    0.2, 0,    0.5, 0.3,   0.7, 0.4, 0.2,  0.3, 0.8, 0.6,   0, 0.2,
                        0.5, 0.3,   0 ],
                                [   10.5, 1.7,  0.3,  4.3, 3.7,  0.8, 1.8, 10.1, 1.4,  2.4, 0.6,   9.3, 2.6, 1.4, 7.7,
                        3.4, 3.5, 1.7, 1.5], 'linear');
        ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

        modulation = (SinOsc.ar([sinenvgen1,sinenvgen2, sinenvgen3, sinenvgen4], 0, ampenvgen));
        process = modulation * reverb;

        env3 = Env.new([0, 0.8254172, 0.8356208, 0],
                        [0.1, 60, 0.1], 'linear');
        envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

        penv = Env.new([0, 0.2, -0.2, -0.5, -0.3, 0, -0.2, 0.5, 0.5, 0],
                                [ 10.5, 18, 9.5, 10.3, 12.7 ], 'linear');
        penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

        Out.ar(out, Pan2.ar(process, penvgen, envgen3))
                }).add;


~w = Buffer.alloc(s, 44100 * 3, 1, 0, 44);

SynthDef(\lowrec23, { arg bufnum = 44;
        var audio, rec;
        audio = (SoundIn.ar([0]));
        rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;
```

```
SynthDef (\lowplay23a, { arg out = 0, amp=0.3, pan=0.55, bufnum = 44 ;
	var audio, reverb, env, envgen;
	audio = PlayBuf.ar(1, bufnum, 0.35);
	reverb = FreeVerb.ar(audio, 0.8, 0.8, 0.8);
	env = Env.new([0, 0.5, 0.5, 0],
			[0.2, 8.17, 0.2 ], 'linear');
	envgen = EnvGen.kr(env, 1.0, doneAction: 2);
	Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\lowplay23b, { arg out = 0, amp=0.3, pan= 0.5, bufnum = 44 ;
	var audio, reverb, env, envgen;
	audio = PlayBuf.ar(1, bufnum, 0.375);
	reverb = FreeVerb.ar(audio, 0.72, 0.72, 0.72);
	env = Env.new([0, 0.5, 0.5, 0],
			[0.2, 7.6, 0.2 ], 'linear');
	envgen = EnvGen.kr(env, 1.0, doneAction: 2);
	Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;


a = Pbind(*[
	instrument:	"wunderkammer23",
	freq:		Pseq([400], 1),
	dur:		Pseq([61], 1),
]);

c = Prout({

	0.1.wait;
	Synth(\lowrec23).postln;
	SystemClock.sched(0.0,{ arg time; time.postln});
	18.57.wait;
	Synth(\lowplay23a).postln;
	11.43.wait;
	18.wait;
	Synth(\lowplay23b).postln;
	20.wait;
	~w.freeMsg

});


~button23 = Ppar([a, c], 1);

);


// end button 23
```

```
//processing activated by button 24
(

SynthDef("wunderkammer24", { arg out=0, amp=0.1, pan=0, freq = 400;
                var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                reverb,
            sinenv1, sinenvgen1, sinenv2, sinenvgen2, sinenv3, sinenvgen3, sinenv4, sinenvgen4,
            modulation, process, ampenv, ampenvgen, penv, penvgen;

        audio = (SoundIn.ar([0]));

        env1 = Env.new([17904.8, 17000, 7050, 15281, 13119, 18000, 19287.2],
                        [11, 14.2, 6, 5, 11, 13], 'linear');
        envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

        filter1 =  BLowPass.ar(audio, envgen1);

        env2 = Env.new([22.6, 59, 600, 457, 11803, 10038, 200, 546, 13.2],
                        [6.2, 7, 23, 5, 4, 6, 5, 4], 'linear');
        envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

        filter2 = BHiPass.ar(filter1, envgen2);

        renv = Env.new([0.08, 0.04],
                        [60.2], 'linear');
        renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

        reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

        sinenv1 = Env.new([2879, 3896, 1167, 4832, 2715, 271, 241 ],
                                [ 7.55, 7.55, 7.55, 7.55, 0.1, 30.05 ], 'sine');
        sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

        sinenv2 = Env.new([2190, 2164, 271, 143, 954, 102, 241 ],
                                [ 30.05, 0.1, 7.55, 7.55, 7.55, 7.55, ], 'sine');
        sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);

        sinenv3 = Env.new([728, 325, 687, 271, 453,  241 ],
                                [ 15.05, 15.05,  0.1, 15.05, 15.05 ], 'sine');
        sinenvgen3 = EnvGen.kr(sinenv3, 1.0, doneAction: 2);

        sinenv4 = Env.new([381, 231, 854, 409, 271, 112, 695, 241 ],
                                [ 10.05, 10.05, 10.05, 0.1, 10.05, 10.05, 10.05 ], 'sine');
        sinenvgen4 = EnvGen.kr(sinenv4, 1.0, doneAction: 2);

        ampenv = Env.new([0, 0.7,   0.6,    0.2, 0.1,    0.4, 0.3,    0.1, 0.4,   0.6, 0.8, 0.2,  0.2, 0.4, 0.2,   0.6,
                        0.1,   0.5, 0.3,   0 ],
                           [   10.5, 1.7,  0.3,  6.3, 3.7,  2.8, 1.8, 4.1, 2.4,  0.4, 1.6,   7.3, 1.6, 1.4, 6.7,
                        3.4, 2.5, 1.7, 8.5], 'linear');
        ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

        modulation = (SinOsc.ar([sinenvgen1,sinenvgen2, sinenvgen3, sinenvgen4], 0, ampenvgen));
        process = modulation * reverb;
```

```
		env3 = Env.new([0, 0.8254172, 0.8356208, 0],
				[0.1, 60, 0.1], 'linear');
		envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

		penv = Env.new([0, -0.5, -0.5, 0.5, -0.3, 0.4, 0.2, 0.8, 0.3, 0],
					[ 12.5, 15, 9.5, 8.3, 15.7 ], 'linear');
		penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

		Out.ar(out, Pan2.ar(process, penvgen, envgen3))
			}).add;

~x = Buffer.alloc(s, 44100 * 10, 1, 0, 47);


SynthDef(\highrec24, { arg bufnum = 47;
		var audio, rec;
		audio = (SoundIn.ar([0]));
		rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;

SynthDef (\highplay24a, { arg out = 0, amp=0.3, pan=0, bufnum = 47 ;
		var audio, reverb, env, envgen;
		audio = PlayBuf.ar(1, bufnum, 1);
		reverb = FreeVerb.ar(audio, 0.02, 0.02, 0.02);
		env = Env.new([0, 0.5, 0.5, 0],
				[0.2, 9.6, 0.2 ], 'linear');
		envgen = EnvGen.kr(env, 1.0, doneAction: 2);
		Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\highplay24b, { arg out = 0, amp=0.3, pan= -0.38, bufnum = 47 ;
		var audio, reverb, env, envgen;
		audio = PlayBuf.ar(1, bufnum, 1.591);
		reverb = FreeVerb.ar(audio, 0.54, 0.54, 0.54);
		env = Env.new([0, 0.5, 0.5, 0],
				[0.2, 5.88, 0.2 ], 'linear');
		envgen = EnvGen.kr(env, 1.0, doneAction: 2);
		Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;




a = Pbind(*[
		instrument:		"wunderkammer24",
		freq:			Pseq([400], 1),
		dur:		Pseq([61], 1),
]);

d = Prout({

		0.1.wait;
		Synth(\highrec24).postln;
		SystemClock.sched(0.0,{ arg time; time.postln});
```

```
            10.wait;
            Synth(\highplay24a).postln;
            20.wait;
            19.02.wait;
            Synth(\highplay24b).postln;
            20.wait;
            ~x.freeMsg

});

~button24 = Ppar([a, d], 1);

);


// end button 24


//processing activated by button 25
(

SynthDef("wunderkammer25", { arg out=0, amp=0.1, pan=0, freq = 400;
                        var audio, filter1, filter2, env1, envgen1, env2, envgen2, env3, envgen3, renv, renvgen,
                                reverb,
                    sinenv1, sinenvgen1, sinenv2, sinenvgen2,
                    modulation, process, ampenv, ampenvgen, penv, penvgen;

            audio = (SoundIn.ar([0]));

            env1 = Env.new([19287.2, 8300, 15193, 7000, 17060, 2115, 12200, 8000, 22000],
                                [10.2, 9, 9, 7, 6, 3, 11, 5], 'linear');
            envgen1 = EnvGen.kr(env1, 1.0, doneAction: 2);

            filter1 =  BLowPass.ar(audio, envgen1);

            env2 = Env.new([13.2, 17, 12, 380, 60, 13, 170, 50, 10],
                                [8.2, 9, 9, 13, 6, 3, 7, 5], 'linear');
            envgen2 = EnvGen.kr(env2, 1.0, doneAction: 2);

            filter2 = BHiPass.ar(filter1, envgen2);

            renv = Env.new([0.04, 0],
                                [60.2], 'linear');
            renvgen = EnvGen.kr(renv, 1.0, doneAction: 2);

            reverb = FreeVerb.ar(filter2, renvgen, renvgen, renvgen);

            sinenv1 = Env.new([1207, 2365, 1112, 546, 321, 699, 452 ],
                                        [ 15.05, 15.05,  0.1, 10.05, 10.05, 10.05], 'sine');
            sinenvgen1 = EnvGen.kr(sinenv1, 1.0, doneAction: 2);

            sinenv2 = Env.new([484, 194, 905, 98, 480, 456, 452 ],
                                        [ 7.55, 7.55, 7.55, 7.55,  0.1, 30.05 ], 'sine');
            sinenvgen2 = EnvGen.kr(sinenv2, 1.0, doneAction: 2);
```

```
        ampenv = Env.new([0, 0.3,   0.5,    0, 0.7,    0, 0.3,    0, 0.3,   0.6, 0.3, 0.4,  0, 0.2, 0.5,   0, 0.2,
                          0.5, 0.5,   0 ],
                            [   8.5, 0.7,  0.3,  6.3, 3.7,  2.8, 1.8, 4.1, 1.4,  0.4, 1.6,   5.3, 1.6, 1.4, 4.7, 2.4,
                          2.5, 10.7, 8.5], 'linear');
        ampenvgen = EnvGen.kr(ampenv, 1.0, doneAction: 2);

        modulation = (SinOsc.ar([sinenvgen1,sinenvgen2], 0, ampenvgen));
        process = modulation * reverb;

        env3 = Env.new([0, 0.8254172, 0.8356208, 0],
                        [0.1, 60, 0.1], 'linear');
        envgen3 = EnvGen.kr(env3, 1.0, doneAction: 2);

        penv = Env.new([0, 0.2, 0.3, -0.4, 0.3, 0.7, -0.2, 0.4, 0.2, 0],
                                [ 3.5, 15, 18.5, 10.3, 13.7 ], 'linear');
        penvgen = EnvGen.kr(penv, 1.0, doneAction: 2);

        Out.ar(out, Pan2.ar(process, penvgen, envgen3))
                }).add;

~y = Buffer.alloc(s, 44100 * 3, 1, 0, 48);

SynthDef(\lowrec25, { arg bufnum = 48;
        var audio, rec;
        audio = (SoundIn.ar([0]));
        rec = RecordBuf.ar(audio, bufnum, doneAction: 2, loop: 0);
}).add;


SynthDef (\lowplay25a, { arg out = 0, amp=0.3, pan= -0.67, bufnum = 48 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 0.313);
        reverb = FreeVerb.ar(audio, 0.94, 0.94, 0.94);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 9.18, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;

SynthDef (\lowplay25b, { arg out = 0, amp=0.3, pan= -0.29, bufnum = 48 ;
        var audio, reverb, env, envgen;
        audio = PlayBuf.ar(1, bufnum, 0.512);
        reverb = FreeVerb.ar(audio, 0.4, 0.4, 0.4);
        env = Env.new([0, 0.5, 0.5, 0],
                        [0.2, 5.45, 0.2 ], 'linear');
        envgen = EnvGen.kr(env, 1.0, doneAction: 2);
        Out.ar(out, Pan2.ar(reverb, pan, envgen));
}).add;


a = Pbind(*[
        instrument:         "wunderkammer25",
        freq:               Pseq([400], 1),
        dur:        Pseq([61], 1),
]);
```

```
c = Prout({

	0.1.wait;
	Synth(\lowrec25).postln;
	SystemClock.sched(0.0,{ arg time; time.postln});
	19.57.wait;
	Synth(\lowplay25a).postln;
	10.43.wait;
	15.86.wait;
	Synth(\lowplay25b).postln;
	20.wait;
	~y.freeMsg

});


~button25 = Ppar([a, c], 1);



);

//processing activated by Series 1 Button

~series1 = Prout({

	0.1.wait;
	"series1".asSymbol.postln;
	SystemClock.sched(0.0,{ arg time; time.postln});
	~button1.play;
	50.wait;
	~button2.play;
	50.wait;
	~button3.play;
	50.wait;
	~button4.play;
	50.wait;
	~button5.play;
	50.wait;
	~button6.play;
	50.wait;
	~button7.play;
	50.wait;
	~button8.play;
	50.wait;
	~button9.play;
	50.wait;
	~button10.play;
	50.wait;
	~button11.play;
	50.wait;
	~button12.play;
	50.wait;
	~button13.play;
```

```
        50.wait;
        ~button14.play;
        50.wait;
        ~button15.play;
        50.wait;
        ~button16.play;
        50.wait;
        ~button17.play;
        50.wait;
        ~button18.play;
        50.wait;
        ~button19.play;
        50.wait;
        ~button20.play;
        50.wait;
        ~button21.play;
        50.wait;
        ~button22.play;
        50.wait;
        ~button23.play;
        50.wait;
        ~button24.play;
        50.wait;
        ~button25.play;

});

//processing activated by Series 2 Button

~series2 = Prout({

0.1.wait;
"series2".asSymbol.postln;
SystemClock.sched(0.0,{ arg time; time.postln});
~button13.play;
50.wait;
~button17.play;
~button9.play;
30.wait;
~button3.play;
30.wait;
~button21.play;
50.wait;
~button18.play;
~button8.play;
30.wait;
~button7.play;
50.wait;
~button23.play;
~button22.play;
50.wait;
~button15.play;
~button19.play;
50.wait;
~button24.play;
```

```
30.wait;
~button2.play;
30.wait;
~button8.play;
~button5.play;
30.wait;
~button4.play;
50.wait;
~button12.play;
~button18.play;
50.wait;
~button16.play;
30.wait;
~button5.play;
30.wait;
~button4.play;
30.wait;
~button20.play;
30.wait;
~button11.play;
~button23.play;
50.wait;
~button6.play;
30.wait;
~button25.play;
~button2.play;
50.wait;
~button21.play;
~button12.play;
50.wait;
~button3.play;
~button17.play;
30.wait;
~button16.play;
30.wait;
~button25.play;
50.wait;
~button1.play;
~button6.play;
30.wait;
~button20.play;
30.wait;
~button14.play;
30.wait;
~button11.play;
50.wait;
~button13.play;
~button24.play;
50.wait;
~button7.play;
30.wait;
~button14.play;
30.wait;
~button9.play;
~button1.play;
```

```
50.wait;
~button19.play;
30.wait;
~button22.play;
~button10.play;
30.wait;
~button15.play;
});

//processing activated by Series 3 Button

~series3 = Prout({

        0.1.wait;
        "series3".asSymbol.postln;
   SystemClock.sched(0.0,{ arg time; time.postln});
        ~button25.play;
        20.wait;
        ~button23.play;
        ~button3.play;
        20.wait;
        ~button9.play;
        20.wait;
        ~button1.play;
   40.wait;
        ~button13.play;
        20.wait;
        ~button17.play;
        20.wait;
        ~button11.play;
        ~button7.play;
        20.wait;
        ~button5.play;
        20.wait;
        ~button21.play;
        20.wait;
        ~button19.play;
        20.wait;
        ~button15.play;

});

//processing activated by Series 4 Button

~series4 = Prout({

        0.1.wait;
        "series4".asSymbol.postln;
   SystemClock.sched(0.0,{ arg time; time.postln});
        ~button16.play;
        20.wait;
        ~button22.play;
        ~button4.play;
        40.wait;
        ~button14.play;
```

```
            ~button20.play;
            20.wait;
            ~button24.play;
    40.wait;
            ~button2.play;
            20.wait;
            ~button18.play;
            ~button10.play;
            40.wait;
            ~button8.play;
            ~button6.play;
            40.wait;
            ~button12.play;

});


//The code that generates of the interface GUI

(
w = Window.new("test1", Rect(3, 500, 1433, 900));
w.view.background_(Color.grey(0.5, 1));


w1 = Button(w, Rect(334*0.7, 20*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~a.alloc(s, 44100 * 3, 1, 0, 0)})
        .action_({ ~button1.play });

w2 = Button(w, Rect(491*0.7, 20*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~b.alloc(s, 44100 * 10, 1, 0, 3)})
        .action_({ ~button2.play });

w3 = Button(w, Rect(648*0.7, 20*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~c.alloc(s, 44100 * 3, 1, 0, 4)})
        .action_({ ~button3.play });

w4 = Button(w, Rect(805*0.7, 20*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~d.alloc(s, 44100 * 10, 1, 0, 7)})
        .action_({ ~button4.play });

w5 = Button(w, Rect(962*0.7, 20*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~e.alloc(s, 44100 * 3, 1, 0, 8)})
        .action_({ ~button5.play });

w6 = Button(w, Rect(1119*0.7, 20*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~f.alloc(s, 44100 * 10, 1, 0, 11)})
        .action_({ ~button6.play });

w7 = Button(w, Rect(1276*0.7, 20*0.7, 140*0.7, 220*0.7))
```

```
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~g.alloc(s, 44100 * 3, 1, 0, 12)})
        .action_({ ~button7.play });

w8 = Button(w, Rect(20*0.7, 260*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~h.alloc(s, 44100 * 10, 1, 0, 15)})
        .action_({ ~button8.play });

w9 = Button(w, Rect(177*0.7, 260*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~i.alloc(s, 44100 * 3, 1, 0, 16)})
        .action_({ ~button9.play });

w10 = Button(w, Rect(334*0.7, 260*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~j.alloc(s, 44100 * 10, 1, 0, 19)})
        .action_({ ~button10.play });

w11 = Button(w, Rect(491*0.7, 260*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~k.alloc(s, 44100 * 3, 1, 0, 20)})
        .action_({ ~button11.play });

w12 = Button(w, Rect(648*0.7, 260*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~l.alloc(s, 44100 * 10, 1, 0, 23)})
        .action_({ ~button12.play });

w13 = Button(w, Rect(805*0.7, 260*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~m.alloc(s, 44100 * 3, 1, 0, 24)})
        .action_({ ~button13.play });

w14 = Button(w, Rect(962*0.7, 260*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~n.alloc(s, 44100 * 10, 1, 0, 27)})
        .action_({ ~button14.play });

w15 = Button(w, Rect(1119*0.7, 260*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~o.alloc(s, 44100 * 3, 1, 0, 28)})
        .action_({ ~button15.play });

w16 = Button(w, Rect(1276*0.7, 260*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~p.alloc(s, 44100 * 10, 1, 0, 31)})
        .action_({ ~button16.play });

w17 = Button(w, Rect(20*0.7, 500*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~q.alloc(s, 44100 * 3, 1, 0, 32)})
        .action_({ ~button17.play });

w18 = Button(w, Rect(177*0.7, 500*0.7, 140*0.7, 220*0.7))
```

```
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~r.alloc(s, 44100 * 10, 1, 0, 35)})
        .action_({ ~button18.play });

w19 = Button(w, Rect(334*0.7, 500*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~s.alloc(s, 44100 * 3, 1, 0, 36)})
        .action_({ ~button19.play });

w20 = Button(w, Rect(491*0.7, 500*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~t.alloc(s, 44100 * 10, 1, 0, 39)})
        .action_({ ~button20.play });

w21 = Button(w, Rect(648*0.7, 500*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~u.alloc(s, 44100 * 3, 1, 0, 40)})
        .action_({ ~button21.play });

w22 = Button(w, Rect(805*0.7, 500*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~v.alloc(s, 44100 * 10, 1, 0, 43)})
        .action_({ ~button22.play });

w23 = Button(w, Rect(962*0.7, 500*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~w.alloc(s, 44100 * 3, 1, 0, 44)})
        .action_({ ~button23.play });

w24 = Button(w, Rect(1119*0.7, 500*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~x.alloc(s, 44100 * 10, 1, 0, 47)})
        .action_({ ~button24.play });

w25 = Button(w, Rect(1276*0.7, 500*0.7, 140*0.7, 220*0.7))
        .states_([["", Color.black, Color.gray(1, 0.4)] ])
        .action_({~y.alloc(s, 44100 * 3, 1, 0, 48)})
        .action_({ ~button25.play });

sw1 = Button(w, Rect(20*0.7, 760*0.7, 460*0.7, 80*0.7))
        .states_([[" ", Color.black, Color.blue(0.5, 0.4)] ])
        .action_({ ~series1.play });

sw2 = Button(w, Rect(20*0.7, 860*0.7, 460*0.7, 80*0.7))
        .states_([[" ", Color.black, Color.blue(0.5, 0.4)] ])
        .action_({ ~series2.play });

sw3 = Button(w, Rect(20*0.7, 960*0.7, 460*0.7, 80*0.7))
        .states_([[" ", Color.black, Color.blue(0.5, 0.4)] ])
        .action_({~a.alloc(s, 44100 * 3, 1, 0, 0)})
        .action_({~c.alloc(s, 44100 * 3, 1, 0, 4)})
        .action_({~e.alloc(s, 44100 * 3, 1, 0, 8)})
        .action_({~g.alloc(s, 44100 * 3, 1, 0, 12)})
        .action_({~i.alloc(s, 44100 * 3, 1, 0, 16)})
        .action_({~k.alloc(s, 44100 * 3, 1, 0, 20)})
```

```
        .action_({~m.alloc(s, 44100 * 3, 1, 0, 24)})
        .action_({~o.alloc(s, 44100 * 3, 1, 0, 28)})
        .action_({~q.alloc(s, 44100 * 3, 1, 0, 32)})
        .action_({~s.alloc(s, 44100 * 3, 1, 0, 36)})
        .action_({~u.alloc(s, 44100 * 3, 1, 0, 40)})
        .action_({~w.alloc(s, 44100 * 3, 1, 0, 44)})
        .action_({~y.alloc(s, 44100 * 3, 1, 0, 48)})
        .action_({ ~series3.play });

sw4 = Button(w, Rect(20*0.7, 1060*0.7, 460*0.7, 80*0.7))
        .states_([[" ", Color.black, Color.blue(0.5, 0.4)] ])
        .action_({~b.alloc(s, 44100 * 10, 1, 0, 3)})
        .action_({~d.alloc(s, 44100 * 10, 1, 0, 7)})
        .action_({~f.alloc(s, 44100 * 10, 1, 0, 11)})
        .action_({~h.alloc(s, 44100 * 10, 1, 0, 15)})
        .action_({~j.alloc(s, 44100 * 10, 1, 0, 19)})
        .action_({~l.alloc(s, 44100 * 10, 1, 0, 23)})
        .action_({~n.alloc(s, 44100 * 10, 1, 0, 27)})
        .action_({~p.alloc(s, 44100 * 10, 1, 0, 31)})
        .action_({~r.alloc(s, 44100 * 10, 1, 0, 35)})
        .action_({~t.alloc(s, 44100 * 10, 1, 0, 39)})
        .action_({~v.alloc(s, 44100 * 10, 1, 0, 43)})
        .action_({~x.alloc(s, 44100 * 10, 1, 0, 47)})
        .action_({ ~series4.play });

w.drawFunc = {

            Pen.fillColor = Color.black;
            Pen.font = Font("Birch Std", 55);
            Pen.stringAtPoint("The ", (15@15));
            Pen.stringAtPoint("Wunderkammer ", (15@70));
            Pen.stringAtPoint("Interface", (15@125));

            Pen.strokeColor = Color.black;
            Pen.moveTo(14@532);
            Pen.lineTo(336@532);
    Pen.lineTo(336@588);
            Pen.lineTo(14@588);
    Pen.lineTo(14@532);
            Pen.stroke;

            Pen.moveTo(14@602);
            Pen.lineTo(336@602);
    Pen.lineTo(336@658);
            Pen.lineTo(14@658);
    Pen.lineTo(14@602);
            Pen.stroke;

            Pen.moveTo(14@672);
            Pen.lineTo(336@672);
    Pen.lineTo(336@728);
            Pen.lineTo(14@728);
    Pen.lineTo(14@672);
            Pen.stroke;
```

```
        Pen.moveTo(14@742);
        Pen.lineTo(336@742);
Pen.lineTo(336@798);
        Pen.lineTo(14@798);
Pen.lineTo(14@742);
        Pen.stroke;


        Pen.fillColor = Color.gray(0.1, 0.4);
        Pen.font = Font("Birch Std", 55);
        Pen.stringAtPoint("Series One ", (100@540));
        Pen.stringAtPoint("Series Two ", (100@610));
        Pen.stringAtPoint("Series Three", (100@680));
        Pen.stringAtPoint("Series Four", (100@750));


        Pen.fillColor = Color.black;
        Pen.font = Font("Arial", 12);
        Pen.stringAtPoint("A series of modules in which the band pass filter progresses", (350@546));
        Pen.stringAtPoint("from the most narrow band to the widest band. ~21 minutes", (350@561));

        Pen.stringAtPoint("A series of overlapping modules in which the sine wave modulation
        progresses", (350@606));
        Pen.stringAtPoint("from a wide spread of four frequencies that converge to a single frequency",
        (350@621));
        Pen.stringAtPoint("that descends to 0 Htz.  ~23 minutes", (350@636));

        Pen.stringAtPoint("A series of overlapping modules in which a 3 second sample is", (350@682));
        Pen.stringAtPoint("played back at rates that move from 0.3 times the rate to real time. ~5
        minutes", (350@697));

        Pen.stringAtPoint("A series of overlapping modules in which a 10 second sample is",
        (350@749));
        Pen.stringAtPoint("played back at rates that move from 3 times the rate to real time. ~5 minutes",
        (350@764));

        Pen.font = Font("Arial", 20);
        Pen.stringAtPoint("Thank you for using the Wunderkammer Interface.",(850@535));
        Pen.font = Font("Arial", 14);
        Pen.stringAtPoint("This project is a live processing environment that is activated by",(850@565));
        Pen.stringAtPoint("pressing the white buttons above, or the blue buttons to the left.",(850@581));
        Pen.stringAtPoint("Each white button triggers a unique, minute-long, processing scheme",
        (850@597));
        Pen.stringAtPoint("based on changes in the contours of a band-pass filter, and shifting",
        (850@613));
        Pen.stringAtPoint("amplitude modulations. The button guide above describes how this",
        (850@631));
        Pen.stringAtPoint("processing scheme is graphically represented for the these buttons.",
        (850@647));
        Pen.stringAtPoint("Please note, that these representations are not meant to be an exact",
        (850@663));
        Pen.stringAtPoint("or specific graph of all the processing parameters that occur, but an",
        (850@679));
        Pen.stringAtPoint("identifiable depiction of the changing sonic characteristics for each",
        (850@695));
```

```
        Pen.stringAtPoint("button. The blue buttons to the left trigger multiple modules that",(850@711));
        Pen.stringAtPoint("are executed in sequence as a chain of events. Each series triggers a",
        (850@727));
        Pen.stringAtPoint("different type of progression inherent in the combined
        processing",(850@743));
        Pen.stringAtPoint("schemes as described.",(850@759));



        Pen.scale(0.7, 0.7);

//button 1

        Pen.strokeColor = Color.black;
        Pen.translate(334,20);


        Pen.moveTo(0@0);
        Pen.lineTo(140@0);
        Pen.lineTo(140@220);
        Pen.lineTo(0@220);
        Pen.lineTo(0@0);
        Pen.stroke;

        Pen.moveTo(10@10);
   Pen.lineTo(10@210);
        Pen.lineTo(130@210);
   Pen.stroke;

        Pen.moveTo(5@10);
        Pen.lineTo(10@10);
        Pen.stroke;
        Pen.moveTo(5@20);
        Pen.lineTo(10@20);
        Pen.stroke;
        Pen.moveTo(5@30);
        Pen.lineTo(10@30);
        Pen.stroke;
        Pen.moveTo(5@40);
        Pen.lineTo(10@40);
        Pen.stroke;
        Pen.moveTo(5@50);
        Pen.lineTo(10@50);
        Pen.stroke;
        Pen.moveTo(5@60);
        Pen.lineTo(10@60);
        Pen.stroke;
        Pen.moveTo(5@70);
        Pen.lineTo(10@70);
        Pen.stroke;
        Pen.moveTo(5@80);
        Pen.lineTo(10@80);
        Pen.stroke;
        Pen.moveTo(5@90);
        Pen.lineTo(10@90);
```

```
Pen.stroke;
Pen.moveTo(5@100);
Pen.lineTo(10@100);
Pen.stroke;
Pen.moveTo(5@110);
Pen.lineTo(10@110);
Pen.stroke;
Pen.moveTo(5@120);
Pen.lineTo(10@120);
Pen.stroke;
Pen.moveTo(5@130);
Pen.lineTo(10@130);
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
Pen.stroke;
Pen.moveTo(30@210);
Pen.lineTo(30@215);
Pen.stroke;
Pen.moveTo(40@210);
Pen.lineTo(40@215);
Pen.stroke;
Pen.moveTo(50@210);
Pen.lineTo(50@215);
Pen.stroke;
Pen.moveTo(60@210);
Pen.lineTo(60@215);
Pen.stroke;
Pen.moveTo(70@210);
Pen.lineTo(70@215);
Pen.stroke;
Pen.moveTo(80@210);
Pen.lineTo(80@215);
```

```
        Pen.stroke;
        Pen.moveTo(90@210);
        Pen.lineTo(90@215);
        Pen.stroke;
        Pen.moveTo(100@210);
        Pen.lineTo(100@215);
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@152.7);
        Pen.lineTo(60.6@152);
        Pen.lineTo(64@46.5);
        Pen.lineTo(71@94.5);
        Pen.lineTo(83.4@152);
Pen.lineTo(130@153.7);
Pen.lineTo(130@151.5);
        Pen.lineTo(87.4@153.2);
        Pen.lineTo(71@185.5);
        Pen.lineTo(60@204.7);
        Pen.lineTo(54.6@152.5);
Pen.lineTo(10@152.5);
        Pen.fill;

        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@182.5);
        Pen.lineTo(70@183.7);
        Pen.stroke;
        Pen.moveTo(70@194.1);
        Pen.lineTo(130@195.8);
        Pen.stroke;


        Pen.fillColor = Color.blue(1, 0.9);
Pen.moveTo(10@10);

Pen.lineTo(16@10);
Pen.lineTo(16@20);
Pen.lineTo(10@20);
Pen.lineTo(10@10);
        Pen.fill;

        Pen.fillColor = Color.red(1, 0.9);
Pen.moveTo(48.86@10);
```

```
    Pen.lineTo(64.56@10);
    Pen.lineTo(64.56@20);
    Pen.lineTo(48.86@20);
    Pen.lineTo(48.86@10);
            Pen.fill;

            Pen.moveTo(102.86@10);
    Pen.lineTo(125.7@10);
    Pen.lineTo(125.7@20);
    Pen.lineTo(102.86@20);
    Pen.lineTo(102.86@10);
            Pen.fill;

            Pen.fillColor = Color.gray(0.1, 0.4);
            Pen.font = Font("Birch Std", 45);
            Pen.stringAtPoint("1", (104@40));

//button 2

            Pen.translate(157,0);
            Pen.strokeColor = Color.black;

            Pen.moveTo(0@0);
            Pen.lineTo(140@0);
            Pen.lineTo(140@220);
            Pen.lineTo(0@220);
            Pen.lineTo(0@0);
            Pen.stroke;

            Pen.moveTo(10@10);
    Pen.lineTo(10@210);
            Pen.lineTo(130@210);
    Pen.stroke;

            Pen.moveTo(5@10);
            Pen.lineTo(10@10);
            Pen.stroke;
            Pen.moveTo(5@20);
            Pen.lineTo(10@20);
            Pen.stroke;
            Pen.moveTo(5@30);
            Pen.lineTo(10@30);
            Pen.stroke;
            Pen.moveTo(5@40);
            Pen.lineTo(10@40);
            Pen.stroke;
            Pen.moveTo(5@50);
            Pen.lineTo(10@50);
            Pen.stroke;
            Pen.moveTo(5@60);
            Pen.lineTo(10@60);
            Pen.stroke;
            Pen.moveTo(5@70);
            Pen.lineTo(10@70);
            Pen.stroke;
```

```
Pen.moveTo(5@80);
Pen.lineTo(10@80);
Pen.stroke;
Pen.moveTo(5@90);
Pen.lineTo(10@90);
Pen.stroke;
Pen.moveTo(5@100);
Pen.lineTo(10@100);
Pen.stroke;
Pen.moveTo(5@110);
Pen.lineTo(10@110);
Pen.stroke;
Pen.moveTo(5@120);
Pen.lineTo(10@120);
Pen.stroke;
Pen.moveTo(5@130);
Pen.lineTo(10@130);
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
Pen.stroke;
Pen.moveTo(30@210);
Pen.lineTo(30@215);
Pen.stroke;
Pen.moveTo(40@210);
Pen.lineTo(40@215);
Pen.stroke;
Pen.moveTo(50@210);
Pen.lineTo(50@215);
Pen.stroke;
Pen.moveTo(60@210);
Pen.lineTo(60@215);
Pen.stroke;
```

```
        Pen.moveTo(70@210);
        Pen.lineTo(70@215);
        Pen.stroke;
        Pen.moveTo(80@210);
        Pen.lineTo(80@215);
        Pen.stroke;
        Pen.moveTo(90@210);
        Pen.lineTo(90@215);
        Pen.stroke;
        Pen.moveTo(100@210);
        Pen.lineTo(100@215);
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@153.7);
        Pen.lineTo(34@178);
        Pen.lineTo(80.2@174);
        Pen.lineTo(92.2@187.8);
        Pen.lineTo(104.4@165);
Pen.lineTo(130@157.2);
Pen.lineTo(130@150.7);
        Pen.lineTo(96.4@139.4);
        Pen.lineTo(80@102);
        Pen.lineTo(58@151.5);
Pen.lineTo(10@151.7);
        Pen.fill;

        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@142.8);
        Pen.lineTo(70@145.2);
        Pen.stroke;
        Pen.moveTo(70@166.3);
        Pen.lineTo(130@166.6);
        Pen.stroke;
        Pen.moveTo(10@150);
        Pen.lineTo(70@150.5);
        Pen.stroke;
        Pen.moveTo(10@176.2);
        Pen.lineTo(70@177.3);
        Pen.stroke;
        Pen.moveTo(70@181.8);
        Pen.lineTo(130@182.2);
        Pen.stroke;
```

```
            Pen.moveTo(10@182.8);
            Pen.lineTo(70@183);
            Pen.stroke;


            Pen.fillColor = Color.blue(1, 0.9);
    Pen.moveTo(10@10);

    Pen.lineTo(30@10);
    Pen.lineTo(30@20);
    Pen.lineTo(10@20);
    Pen.lineTo(10@10);
            Pen.fill;

            Pen.fillColor = Color.red(1, 0.9);
    Pen.moveTo(47.34@10);
    Pen.lineTo(70.24@10);
    Pen.lineTo(70.24@20);
    Pen.lineTo(47.34@20);
    Pen.lineTo(47.34@10);
            Pen.fill;

            Pen.moveTo(92.78@10);
    Pen.lineTo(121.64@10);
    Pen.lineTo(121.64@20);
    Pen.lineTo(92.78@20);
    Pen.lineTo(92.78@10);
            Pen.fill;

            Pen.fillColor = Color.gray(0.1, 0.4);
            Pen.font = Font("Birch Std", 45);
            Pen.stringAtPoint("2", (104@40));

//button3

            Pen.translate(157,0);
            Pen.strokeColor = Color.black;
            Pen.moveTo(0@0);
            Pen.lineTo(140@0);
            Pen.lineTo(140@220);
            Pen.lineTo(0@220);
            Pen.lineTo(0@0);
            Pen.stroke;

            Pen.strokeColor = Color.black;
    Pen.moveTo(10@10);
    Pen.lineTo(10@210);
            Pen.lineTo(130@210);
    Pen.stroke;

            Pen.moveTo(5@10);
            Pen.lineTo(10@10);
            Pen.stroke;
            Pen.moveTo(5@20);
            Pen.lineTo(10@20);
```

```
Pen.stroke;
Pen.moveTo(5@30);
Pen.lineTo(10@30);
Pen.stroke;
Pen.moveTo(5@40);
Pen.lineTo(10@40);
Pen.stroke;
Pen.moveTo(5@50);
Pen.lineTo(10@50);
Pen.stroke;
Pen.moveTo(5@60);
Pen.lineTo(10@60);
Pen.stroke;
Pen.moveTo(5@70);
Pen.lineTo(10@70);
Pen.stroke;
Pen.moveTo(5@80);
Pen.lineTo(10@80);
Pen.stroke;
Pen.moveTo(5@90);
Pen.lineTo(10@90);
Pen.stroke;
Pen.moveTo(5@100);
Pen.lineTo(10@100);
Pen.stroke;
Pen.moveTo(5@110);
Pen.lineTo(10@110);
Pen.stroke;
Pen.moveTo(5@120);
Pen.lineTo(10@120);
Pen.stroke;
Pen.moveTo(5@130);
Pen.lineTo(10@130);
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;
```

```
        Pen.moveTo(20@210);
        Pen.lineTo(20@215);
        Pen.stroke;
        Pen.moveTo(30@210);
        Pen.lineTo(30@215);
        Pen.stroke;
        Pen.moveTo(40@210);
        Pen.lineTo(40@215);
        Pen.stroke;
        Pen.moveTo(50@210);
        Pen.lineTo(50@215);
        Pen.stroke;
        Pen.moveTo(60@210);
        Pen.lineTo(60@215);
        Pen.stroke;
        Pen.moveTo(70@210);
        Pen.lineTo(70@215);
        Pen.stroke;
        Pen.moveTo(80@210);
        Pen.lineTo(80@215);
        Pen.stroke;
        Pen.moveTo(90@210);
        Pen.lineTo(90@215);
        Pen.stroke;
        Pen.moveTo(100@210);
        Pen.lineTo(100@215);
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@157.2);
Pen.lineTo(130@159.5);
Pen.lineTo(130@148.7);
        Pen.lineTo(104.4@148);
        Pen.lineTo(104@110);
        Pen.lineTo(90@68);
        Pen.lineTo(42@139.4);
        Pen.lineTo(30@149.9);
Pen.lineTo(10@150.7);
        Pen.fill;

        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@172.7);
```

```
        Pen.lineTo(70@175);
        Pen.stroke;
        Pen.moveTo(70@123.2);
        Pen.lineTo(130@124.8);
        Pen.stroke;
        Pen.moveTo(70@146);
        Pen.lineTo(130@146.3);
        Pen.stroke;
        Pen.moveTo(10@182.1);
        Pen.lineTo(70@182.4);
        Pen.stroke;
        Pen.moveTo(70@166.4);
        Pen.lineTo(130@167.2);
        Pen.stroke;
        Pen.moveTo(70@196.9);
        Pen.lineTo(130@197.2);
        Pen.stroke;


        Pen.fillColor = Color.blue(1, 0.9);
    Pen.moveTo(10@10);

    Pen.lineTo(16@10);
    Pen.lineTo(16@20);
    Pen.lineTo(10@20);
    Pen.lineTo(10@10);
        Pen.fill;

        Pen.fillColor = Color.red(1, 0.9);
    Pen.moveTo(45.72@10);
    Pen.lineTo(71.46@10);
    Pen.lineTo(71.46@20);
    Pen.lineTo(45.72@20);
    Pen.lineTo(45.72@10);
        Pen.fill;

        Pen.moveTo(105.14@10);
    Pen.lineTo(130.3@10);
    Pen.lineTo(130.3@20);
    Pen.lineTo(105.14@20);
    Pen.lineTo(105.14@10);
        Pen.fill;

        Pen.fillColor = Color.gray(0.1, 0.4);
        Pen.font = Font("Birch Std", 45);
        Pen.stringAtPoint("3", (104@40));

//button 4

        Pen.translate(157,0);
        Pen.strokeColor = Color.black;

        Pen.moveTo(0@0);
        Pen.lineTo(140@0);
        Pen.lineTo(140@220);
```

```
        Pen.lineTo(0@220);
        Pen.lineTo(0@0);
        Pen.stroke;

        Pen.moveTo(10@10);
Pen.lineTo(10@210);
        Pen.lineTo(130@210);
Pen.stroke;

        Pen.moveTo(5@10);
        Pen.lineTo(10@10);
        Pen.stroke;
        Pen.moveTo(5@20);
        Pen.lineTo(10@20);
        Pen.stroke;
        Pen.moveTo(5@30);
        Pen.lineTo(10@30);
        Pen.stroke;
        Pen.moveTo(5@40);
        Pen.lineTo(10@40);
        Pen.stroke;
        Pen.moveTo(5@50);
        Pen.lineTo(10@50);
        Pen.stroke;
        Pen.moveTo(5@60);
        Pen.lineTo(10@60);
        Pen.stroke;
        Pen.moveTo(5@70);
        Pen.lineTo(10@70);
        Pen.stroke;
        Pen.moveTo(5@80);
        Pen.lineTo(10@80);
        Pen.stroke;
        Pen.moveTo(5@90);
        Pen.lineTo(10@90);
        Pen.stroke;
        Pen.moveTo(5@100);
        Pen.lineTo(10@100);
        Pen.stroke;
        Pen.moveTo(5@110);
        Pen.lineTo(10@110);
        Pen.stroke;
        Pen.moveTo(5@120);
        Pen.lineTo(10@120);
        Pen.stroke;
        Pen.moveTo(5@130);
        Pen.lineTo(10@130);
        Pen.stroke;
        Pen.moveTo(5@140);
        Pen.lineTo(10@140);
        Pen.stroke;
        Pen.moveTo(5@150);
        Pen.lineTo(10@150);
        Pen.stroke;
        Pen.moveTo(5@160);
```

```
        Pen.lineTo(10@160);
        Pen.stroke;
        Pen.moveTo(5@170);
        Pen.lineTo(10@170);
        Pen.stroke;
        Pen.moveTo(5@180);
        Pen.lineTo(10@180);
        Pen.stroke;
        Pen.moveTo(5@190);
        Pen.lineTo(10@190);
        Pen.stroke;
        Pen.moveTo(5@200);
        Pen.lineTo(10@200);
        Pen.stroke;

        Pen.moveTo(20@210);
        Pen.lineTo(20@215);
        Pen.stroke;
        Pen.moveTo(30@210);
        Pen.lineTo(30@215);
        Pen.stroke;
        Pen.moveTo(40@210);
        Pen.lineTo(40@215);
        Pen.stroke;
        Pen.moveTo(50@210);
        Pen.lineTo(50@215);
        Pen.stroke;
        Pen.moveTo(60@210);
        Pen.lineTo(60@215);
        Pen.stroke;
        Pen.moveTo(70@210);
        Pen.lineTo(70@215);
        Pen.stroke;
        Pen.moveTo(80@210);
        Pen.lineTo(80@215);
        Pen.stroke;
        Pen.moveTo(90@210);
        Pen.lineTo(90@215);
        Pen.stroke;
        Pen.moveTo(100@210);
        Pen.lineTo(100@215);
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);
```

```
Pen.lineTo(10@159.5);
	Pen.lineTo(50@184);
	Pen.lineTo(56@156.3);
	Pen.lineTo(84@191.7);
	Pen.lineTo(98@187);
	Pen.lineTo(126@152);
Pen.lineTo(130@161.2);
Pen.lineTo(130@146.3);
Pen.lineTo(10@148.7);
	Pen.fill;

	Pen.strokeColor = Color.magenta(0.3, 0.9);
	Pen.width = 3;
	Pen.moveTo(10@157.2);
	Pen.lineTo(70@158.5);
	Pen.stroke;
	Pen.moveTo(70@149.2);
	Pen.lineTo(130@151.3);
	Pen.stroke;
	Pen.moveTo(70@151.5);
	Pen.lineTo(130@151.9);
	Pen.stroke;
	Pen.moveTo(10@181);
	Pen.lineTo(70@181.2);
	Pen.stroke;
	Pen.moveTo(70@179.3);
	Pen.lineTo(130@180.5);
	Pen.stroke;
	Pen.moveTo(70@181.5);
	Pen.lineTo(130@182.5);
	Pen.stroke;


	Pen.fillColor = Color.blue(1, 0.9);
Pen.moveTo(10@10);

Pen.lineTo(30@10);
Pen.lineTo(30@20);
Pen.lineTo(10@20);
Pen.lineTo(10@10);
	Pen.fill;

	Pen.fillColor = Color.red(1, 0.9);
Pen.moveTo(45.96@10);
Pen.lineTo(69.4@10);
Pen.lineTo(69.4@20);
Pen.lineTo(45.96@20);
Pen.lineTo(45.96@10);
	Pen.fill;

	Pen.moveTo(114.98@10);
Pen.lineTo(134.7@10);
Pen.lineTo(134.7@20);
Pen.lineTo(114.98@20);
Pen.lineTo(114.98@10);
```

```
        Pen.fill;

        Pen.fillColor = Color.gray(0.1, 0.4);
        Pen.font = Font("Birch Std", 45);
        Pen.stringAtPoint("4", (104@40));

//button 5

        Pen.translate(157,0);
        Pen.strokeColor = Color.black;

        Pen.moveTo(0@0);
        Pen.lineTo(140@0);
        Pen.lineTo(140@220);
        Pen.lineTo(0@220);
        Pen.lineTo(0@0);
        Pen.stroke;

        Pen.moveTo(10@10);
    Pen.lineTo(10@210);
        Pen.lineTo(130@210);
    Pen.stroke;

        Pen.moveTo(5@10);
        Pen.lineTo(10@10);
        Pen.stroke;
        Pen.moveTo(5@20);
        Pen.lineTo(10@20);
        Pen.stroke;
        Pen.moveTo(5@30);
        Pen.lineTo(10@30);
        Pen.stroke;
        Pen.moveTo(5@40);
        Pen.lineTo(10@40);
        Pen.stroke;
        Pen.moveTo(5@50);
        Pen.lineTo(10@50);
        Pen.stroke;
        Pen.moveTo(5@60);
        Pen.lineTo(10@60);
        Pen.stroke;
        Pen.moveTo(5@70);
        Pen.lineTo(10@70);
        Pen.stroke;
        Pen.moveTo(5@80);
        Pen.lineTo(10@80);
        Pen.stroke;
        Pen.moveTo(5@90);
        Pen.lineTo(10@90);
        Pen.stroke;
        Pen.moveTo(5@100);
        Pen.lineTo(10@100);
        Pen.stroke;
        Pen.moveTo(5@110);
        Pen.lineTo(10@110);
```

```
Pen.stroke;
Pen.moveTo(5@120);
Pen.lineTo(10@120);
Pen.stroke;
Pen.moveTo(5@130);
Pen.lineTo(10@130);
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
Pen.stroke;
Pen.moveTo(30@210);
Pen.lineTo(30@215);
Pen.stroke;
Pen.moveTo(40@210);
Pen.lineTo(40@215);
Pen.stroke;
Pen.moveTo(50@210);
Pen.lineTo(50@215);
Pen.stroke;
Pen.moveTo(60@210);
Pen.lineTo(60@215);
Pen.stroke;
Pen.moveTo(70@210);
Pen.lineTo(70@215);
Pen.stroke;
Pen.moveTo(80@210);
Pen.lineTo(80@215);
Pen.stroke;
Pen.moveTo(90@210);
Pen.lineTo(90@215);
Pen.stroke;
Pen.moveTo(100@210);
Pen.lineTo(100@215);
```

```
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@161.2);
        Pen.lineTo(30@166.8);
        Pen.lineTo(36.2@155.8);
        Pen.lineTo(52.2@163.2);
        Pen.lineTo(104.4@191.8);
Pen.lineTo(130@164);
Pen.lineTo(130@142.8);
        Pen.lineTo(81.4@152);
        Pen.lineTo(67@36.3);
        Pen.lineTo(46@110);
        Pen.lineTo(24.6@152.3);
Pen.lineTo(10@146.3);
        Pen.fill;

        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@155.5);
        Pen.lineTo(70@156.5);
        Pen.stroke;
        Pen.moveTo(70@146.9);
        Pen.lineTo(130@148.3);
        Pen.stroke;
        Pen.moveTo(70@150.5);
        Pen.lineTo(130@151.5);
        Pen.stroke;
        Pen.moveTo(10@180.8);
        Pen.lineTo(70@181);
        Pen.stroke;
        Pen.moveTo(70@178.2);
        Pen.lineTo(130@179.3);
        Pen.stroke;
        Pen.moveTo(70@182.2);
        Pen.lineTo(130@183.5);
        Pen.stroke;

        Pen.fillColor = Color.blue(1, 0.9);
Pen.moveTo(10@10);

Pen.lineTo(16@10);
Pen.lineTo(16@20);
Pen.lineTo(10@20);
```

```
    Pen.lineTo(10@10);
            Pen.fill;

            Pen.fillColor = Color.red(1, 0.9);
    Pen.moveTo(41.14@10);
    Pen.lineTo(62.28@10);
    Pen.lineTo(62.28@20);
    Pen.lineTo(41.14@20);
    Pen.lineTo(41.14@10);
            Pen.fill;

            Pen.moveTo(99.72@10);
    Pen.lineTo(119.42@10);
    Pen.lineTo(119.42@20);
    Pen.lineTo(99.72@20);
    Pen.lineTo(99.72@10);
            Pen.fill;

            Pen.fillColor = Color.gray(0.1, 0.4);
            Pen.font = Font("Birch Std", 45);
            Pen.stringAtPoint("5", (104@40));

//button 6

            Pen.translate(157,0);
            Pen.strokeColor = Color.black;

            Pen.moveTo(0@0);
            Pen.lineTo(140@0);
            Pen.lineTo(140@220);
            Pen.lineTo(0@220);
            Pen.lineTo(0@0);
            Pen.stroke;

            Pen.moveTo(10@10);
    Pen.lineTo(10@210);
            Pen.lineTo(130@210);
    Pen.stroke;

            Pen.moveTo(5@10);
            Pen.lineTo(10@10);
            Pen.stroke;
            Pen.moveTo(5@20);
            Pen.lineTo(10@20);
            Pen.stroke;
            Pen.moveTo(5@30);
            Pen.lineTo(10@30);
            Pen.stroke;
            Pen.moveTo(5@40);
            Pen.lineTo(10@40);
            Pen.stroke;
            Pen.moveTo(5@50);
            Pen.lineTo(10@50);
            Pen.stroke;
            Pen.moveTo(5@60);
```

```
Pen.lineTo(10@60);
Pen.stroke;
Pen.moveTo(5@70);
Pen.lineTo(10@70);
Pen.stroke;
Pen.moveTo(5@80);
Pen.lineTo(10@80);
Pen.stroke;
Pen.moveTo(5@90);
Pen.lineTo(10@90);
Pen.stroke;
Pen.moveTo(5@100);
Pen.lineTo(10@100);
Pen.stroke;
Pen.moveTo(5@110);
Pen.lineTo(10@110);
Pen.stroke;
Pen.moveTo(5@120);
Pen.lineTo(10@120);
Pen.stroke;
Pen.moveTo(5@130);
Pen.lineTo(10@130);
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
Pen.stroke;
Pen.moveTo(30@210);
Pen.lineTo(30@215);
Pen.stroke;
Pen.moveTo(40@210);
Pen.lineTo(40@215);
Pen.stroke;
Pen.moveTo(50@210);
```

```
        Pen.lineTo(50@215);
        Pen.stroke;
        Pen.moveTo(60@210);
        Pen.lineTo(60@215);
        Pen.stroke;
        Pen.moveTo(70@210);
        Pen.lineTo(70@215);
        Pen.stroke;
        Pen.moveTo(80@210);
        Pen.lineTo(80@215);
        Pen.stroke;
        Pen.moveTo(90@210);
        Pen.lineTo(90@215);
        Pen.stroke;
        Pen.moveTo(100@210);
        Pen.lineTo(100@215);
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@164);
        Pen.lineTo(44@150.4);
        Pen.lineTo(70.2@193);
        Pen.lineTo(72.2@187.3);
        Pen.lineTo(104.4@165.1);
Pen.lineTo(130@166.2);
Pen.lineTo(130@138.5);
Pen.lineTo(124.4@148);
        Pen.lineTo(84@110);
        Pen.lineTo(50@68.2);
        Pen.lineTo(42@62.8);
        Pen.lineTo(12@149.7);
        Pen.lineTo(10@142.8);
        Pen.fill;

        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@163.5);
        Pen.lineTo(70@165.2);
        Pen.stroke;
        Pen.moveTo(70@183.5);
        Pen.lineTo(130@184.3);
        Pen.stroke;
        Pen.moveTo(10@181.8);
        Pen.lineTo(70@182.3);
```

```
        Pen.stroke;

        Pen.fillColor = Color.blue(1, 0.9);
    Pen.moveTo(10@10);

    Pen.lineTo(30@10);
    Pen.lineTo(30@20);
    Pen.lineTo(10@20);
    Pen.lineTo(10@10);
        Pen.fill;

        Pen.fillColor = Color.red(1, 0.9);
    Pen.moveTo(41.8@10);
    Pen.lineTo(76.58@10);
    Pen.lineTo(76.58@20);
    Pen.lineTo(41.8@20);
    Pen.lineTo(41.8@10);
        Pen.fill;

        Pen.moveTo(93.46@10);
    Pen.lineTo(122.04@10);
    Pen.lineTo(122.04@20);
    Pen.lineTo(93.46@20);
    Pen.lineTo(93.46@10);
        Pen.fill;

        Pen.fillColor = Color.gray(0.1, 0.4);
        Pen.font = Font("Birch Std", 45);
        Pen.stringAtPoint("6", (104@40));

// button 7

        Pen.translate(157,0);
        Pen.strokeColor = Color.black;

        Pen.moveTo(0@0);
        Pen.lineTo(140@0);
        Pen.lineTo(140@220);
        Pen.lineTo(0@220);
        Pen.lineTo(0@0);
        Pen.stroke;

        Pen.moveTo(10@10);
    Pen.lineTo(10@210);
        Pen.lineTo(130@210);
    Pen.stroke;

        Pen.moveTo(5@10);
        Pen.lineTo(10@10);
        Pen.stroke;
        Pen.moveTo(5@20);
        Pen.lineTo(10@20);
        Pen.stroke;
        Pen.moveTo(5@30);
        Pen.lineTo(10@30);
```

```
Pen.stroke;
Pen.moveTo(5@40);
Pen.lineTo(10@40);
Pen.stroke;
Pen.moveTo(5@50);
Pen.lineTo(10@50);
Pen.stroke;
Pen.moveTo(5@60);
Pen.lineTo(10@60);
Pen.stroke;
Pen.moveTo(5@70);
Pen.lineTo(10@70);
Pen.stroke;
Pen.moveTo(5@80);
Pen.lineTo(10@80);
Pen.stroke;
Pen.moveTo(5@90);
Pen.lineTo(10@90);
Pen.stroke;
Pen.moveTo(5@100);
Pen.lineTo(10@100);
Pen.stroke;
Pen.moveTo(5@110);
Pen.lineTo(10@110);
Pen.stroke;
Pen.moveTo(5@120);
Pen.lineTo(10@120);
Pen.stroke;
Pen.moveTo(5@130);
Pen.lineTo(10@130);
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
```

```
        Pen.stroke;
        Pen.moveTo(30@210);
        Pen.lineTo(30@215);
        Pen.stroke;
        Pen.moveTo(40@210);
        Pen.lineTo(40@215);
        Pen.stroke;
        Pen.moveTo(50@210);
        Pen.lineTo(50@215);
        Pen.stroke;
        Pen.moveTo(60@210);
        Pen.lineTo(60@215);
        Pen.stroke;
        Pen.moveTo(70@210);
        Pen.lineTo(70@215);
        Pen.stroke;
        Pen.moveTo(80@210);
        Pen.lineTo(80@215);
        Pen.stroke;
        Pen.moveTo(90@210);
        Pen.lineTo(90@215);
        Pen.stroke;
        Pen.moveTo(100@210);
        Pen.lineTo(100@215);
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@166.2);
Pen.lineTo(130@168.5);
Pen.lineTo(130@134.5);
Pen.lineTo(10@138.5);
        Pen.fill;

        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@189.9);
        Pen.lineTo(70@191.2);
        Pen.stroke;
        Pen.moveTo(70@131.4);
        Pen.lineTo(130@133.2);
        Pen.stroke;
        Pen.moveTo(70@147.7);
        Pen.lineTo(130@148.2);
        Pen.stroke;
```

```
		Pen.moveTo(70@170.5);
		Pen.lineTo(130@171.2);
		Pen.stroke;
		Pen.moveTo(70@189.8);
		Pen.lineTo(130@190.7);
		Pen.stroke;

		Pen.fillColor = Color.blue(1, 0.9);
	Pen.moveTo(10@10);

	Pen.lineTo(16@10);
	Pen.lineTo(16@20);
	Pen.lineTo(10@20);
	Pen.lineTo(10@10);
		Pen.fill;

		Pen.fillColor = Color.red(1, 0.9);
	Pen.moveTo(48.28@10);
	Pen.lineTo(76.58@10);
	Pen.lineTo(76.58@20);
	Pen.lineTo(48.28@20);
	Pen.lineTo(48.28@10);
		Pen.fill;

		Pen.moveTo(100.28@10);
	Pen.lineTo(120.58@10);
	Pen.lineTo(120.50@20);
	Pen.lineTo(100.28@20);
	Pen.lineTo(100.28@10);
		Pen.fill;

		Pen.fillColor = Color.gray(0.1, 0.4);
		Pen.font = Font("Birch Std", 45);
		Pen.stringAtPoint("7", (104@40));

		//-1256@280

//button 8

		Pen.translate(-1256,240);
		Pen.strokeColor = Color.black;

		Pen.moveTo(0@0);
		Pen.lineTo(140@0);
		Pen.lineTo(140@220);
		Pen.lineTo(0@220);
		Pen.lineTo(0@0);
		Pen.stroke;

		Pen.moveTo(10@10);
	Pen.lineTo(10@210);
		Pen.lineTo(130@210);
	Pen.stroke;

		Pen.moveTo(5@10);
```

```
Pen.lineTo(10@10);
Pen.stroke;
Pen.moveTo(5@20);
Pen.lineTo(10@20);
Pen.stroke;
Pen.moveTo(5@30);
Pen.lineTo(10@30);
Pen.stroke;
Pen.moveTo(5@40);
Pen.lineTo(10@40);
Pen.stroke;
Pen.moveTo(5@50);
Pen.lineTo(10@50);
Pen.stroke;
Pen.moveTo(5@60);
Pen.lineTo(10@60);
Pen.stroke;
Pen.moveTo(5@70);
Pen.lineTo(10@70);
Pen.stroke;
Pen.moveTo(5@80);
Pen.lineTo(10@80);
Pen.stroke;
Pen.moveTo(5@90);
Pen.lineTo(10@90);
Pen.stroke;
Pen.moveTo(5@100);
Pen.lineTo(10@100);
Pen.stroke;
Pen.moveTo(5@110);
Pen.lineTo(10@110);
Pen.stroke;
Pen.moveTo(5@120);
Pen.lineTo(10@120);
Pen.stroke;
Pen.moveTo(5@130);
Pen.lineTo(10@130);
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
```

```
        Pen.stroke;
        Pen.moveTo(5@200);
        Pen.lineTo(10@200);
        Pen.stroke;

        Pen.moveTo(20@210);
        Pen.lineTo(20@215);
        Pen.stroke;
        Pen.moveTo(30@210);
        Pen.lineTo(30@215);
        Pen.stroke;
        Pen.moveTo(40@210);
        Pen.lineTo(40@215);
        Pen.stroke;
        Pen.moveTo(50@210);
        Pen.lineTo(50@215);
        Pen.stroke;
        Pen.moveTo(60@210);
        Pen.lineTo(60@215);
        Pen.stroke;
        Pen.moveTo(70@210);
        Pen.lineTo(70@215);
        Pen.stroke;
        Pen.moveTo(80@210);
        Pen.lineTo(80@215);
        Pen.stroke;
        Pen.moveTo(90@210);
        Pen.lineTo(90@215);
        Pen.stroke;
        Pen.moveTo(100@210);
        Pen.lineTo(100@215);
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@168.5);
Pen.lineTo(130@170.8);
Pen.lineTo(130@130);
        Pen.lineTo(88.4@148);
        Pen.lineTo(82@46.5);
        Pen.lineTo(48@68.2);
        Pen.lineTo(40@139.4);
        Pen.lineTo(26@141.9);
Pen.lineTo(10@134.5);
        Pen.fill;
```

```
        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@144.7);
        Pen.lineTo(70@147.1);
        Pen.stroke;
        Pen.moveTo(70@129.7);
        Pen.lineTo(130@131.3);
        Pen.stroke;
        Pen.moveTo(10@150.3);
        Pen.lineTo(70@150.7);
        Pen.stroke;
        Pen.moveTo(70@147.5);
        Pen.lineTo(130@147.9);
        Pen.stroke;
        Pen.moveTo(10@177.3);
        Pen.lineTo(70@178.2);
        Pen.stroke;
        Pen.moveTo(70@169.7);
        Pen.lineTo(130@170.4);
        Pen.stroke;
        Pen.moveTo(10@183.4);
        Pen.lineTo(70@183.7);
        Pen.stroke;
        Pen.moveTo(70@191.1);
        Pen.lineTo(130@191.7);
        Pen.stroke;

        Pen.fillColor = Color.blue(1, 0.9);
Pen.moveTo(10@10);

Pen.lineTo(30@10);
Pen.lineTo(30@20);
Pen.lineTo(10@20);
Pen.lineTo(10@10);
        Pen.fill;

        Pen.fillColor = Color.red(1, 0.9);
Pen.moveTo(34.86@10);
Pen.lineTo(62.86@10);
Pen.lineTo(62.86@20);
Pen.lineTo(34.86@20);
Pen.lineTo(34.86@10);
        Pen.fill;

        Pen.moveTo(115.68@10);
Pen.lineTo(135.1@10);
Pen.lineTo(135.1@20);
Pen.lineTo(115.68@20);
Pen.lineTo(115.68@10);
        Pen.fill;

        Pen.fillColor = Color.gray(0.1, 0.4);
        Pen.font = Font("Birch Std", 45);
        Pen.stringAtPoint("8", (104@40));
```

```
// button 9
        Pen.translate(157,0);
        Pen.strokeColor = Color.black;

        Pen.moveTo(0@0);
        Pen.lineTo(140@0);
        Pen.lineTo(140@220);
        Pen.lineTo(0@220);
        Pen.lineTo(0@0);
        Pen.stroke;

        Pen.moveTo(10@10);
    Pen.lineTo(10@210);
        Pen.lineTo(130@210);
    Pen.stroke;

        Pen.moveTo(5@10);
        Pen.lineTo(10@10);
        Pen.stroke;
        Pen.moveTo(5@20);
        Pen.lineTo(10@20);
        Pen.stroke;
        Pen.moveTo(5@30);
        Pen.lineTo(10@30);
        Pen.stroke;
        Pen.moveTo(5@40);
        Pen.lineTo(10@40);
        Pen.stroke;
        Pen.moveTo(5@50);
        Pen.lineTo(10@50);
        Pen.stroke;
        Pen.moveTo(5@60);
        Pen.lineTo(10@60);
        Pen.stroke;
        Pen.moveTo(5@70);
        Pen.lineTo(10@70);
        Pen.stroke;
        Pen.moveTo(5@80);
        Pen.lineTo(10@80);
        Pen.stroke;
        Pen.moveTo(5@90);
        Pen.lineTo(10@90);
        Pen.stroke;
        Pen.moveTo(5@100);
        Pen.lineTo(10@100);
        Pen.stroke;
        Pen.moveTo(5@110);
        Pen.lineTo(10@110);
        Pen.stroke;
        Pen.moveTo(5@120);
        Pen.lineTo(10@120);
        Pen.stroke;
        Pen.moveTo(5@130);
        Pen.lineTo(10@130);
```

```
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
Pen.stroke;
Pen.moveTo(30@210);
Pen.lineTo(30@215);
Pen.stroke;
Pen.moveTo(40@210);
Pen.lineTo(40@215);
Pen.stroke;
Pen.moveTo(50@210);
Pen.lineTo(50@215);
Pen.stroke;
Pen.moveTo(60@210);
Pen.lineTo(60@215);
Pen.stroke;
Pen.moveTo(70@210);
Pen.lineTo(70@215);
Pen.stroke;
Pen.moveTo(80@210);
Pen.lineTo(80@215);
Pen.stroke;
Pen.moveTo(90@210);
Pen.lineTo(90@215);
Pen.stroke;
Pen.moveTo(100@210);
Pen.lineTo(100@215);
Pen.stroke;
Pen.moveTo(110@210);
Pen.lineTo(110@215);
Pen.stroke;
Pen.moveTo(120@210);
Pen.lineTo(120@215);
```

```
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@170.8);
        Pen.lineTo(64@191.2);
        Pen.lineTo(70.2@165.8);
        Pen.lineTo(92.2@171.2);
        Pen.lineTo(104.4@197);
Pen.lineTo(130@173.1);
Pen.lineTo(130@125);
Pen.lineTo(10@130);
        Pen.fill;

        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@192.5);
        Pen.lineTo(70@194.1);
        Pen.stroke;
        Pen.moveTo(70@121.7);
        Pen.lineTo(130@123.2);
        Pen.stroke;
        Pen.moveTo(70@145.5);
        Pen.lineTo(130@146);
        Pen.stroke;
        Pen.moveTo(70@165.7);
        Pen.lineTo(130@166.5);
        Pen.stroke;
        Pen.moveTo(70@198.8);
        Pen.lineTo(130@196.8);
        Pen.stroke;


        Pen.fillColor = Color.blue(1, 0.9);
Pen.moveTo(10@10);

Pen.lineTo(16@10);
Pen.lineTo(16@20);
Pen.lineTo(10@20);
Pen.lineTo(10@10);
        Pen.fill;

        Pen.fillColor = Color.red(1, 0.9);
Pen.moveTo(48.58@10);
Pen.lineTo(77.14@10);
Pen.lineTo(77.14@20);
Pen.lineTo(48.58@20);
Pen.lineTo(48.58@10);
        Pen.fill;

        Pen.moveTo(104.58@10);
```

```
    Pen.lineTo(129.14@10);
    Pen.lineTo(129.14@20);
    Pen.lineTo(104.58@20);
    Pen.lineTo(104.58@10);
            Pen.fill;

            Pen.fillColor = Color.gray(0.1, 0.4);
            Pen.font = Font("Birch Std", 45);
            Pen.stringAtPoint("9", (104@40));

//button 10
            Pen.translate(157,0);
            Pen.strokeColor = Color.black;

            Pen.moveTo(0@0);
            Pen.lineTo(140@0);
            Pen.lineTo(140@220);
            Pen.lineTo(0@220);
            Pen.lineTo(0@0);
            Pen.stroke;

            Pen.moveTo(10@10);
    Pen.lineTo(10@210);
            Pen.lineTo(130@210);
    Pen.stroke;

            Pen.moveTo(5@10);
            Pen.lineTo(10@10);
            Pen.stroke;
            Pen.moveTo(5@20);
            Pen.lineTo(10@20);
            Pen.stroke;
            Pen.moveTo(5@30);
            Pen.lineTo(10@30);
            Pen.stroke;
            Pen.moveTo(5@40);
            Pen.lineTo(10@40);
            Pen.stroke;
            Pen.moveTo(5@50);
            Pen.lineTo(10@50);
            Pen.stroke;
            Pen.moveTo(5@60);
            Pen.lineTo(10@60);
            Pen.stroke;
            Pen.moveTo(5@70);
            Pen.lineTo(10@70);
            Pen.stroke;
            Pen.moveTo(5@80);
            Pen.lineTo(10@80);
            Pen.stroke;
            Pen.moveTo(5@90);
            Pen.lineTo(10@90);
            Pen.stroke;
            Pen.moveTo(5@100);
            Pen.lineTo(10@100);
```

```
Pen.stroke;
Pen.moveTo(5@110);
Pen.lineTo(10@110);
Pen.stroke;
Pen.moveTo(5@120);
Pen.lineTo(10@120);
Pen.stroke;
Pen.moveTo(5@130);
Pen.lineTo(10@130);
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
Pen.stroke;
Pen.moveTo(30@210);
Pen.lineTo(30@215);
Pen.stroke;
Pen.moveTo(40@210);
Pen.lineTo(40@215);
Pen.stroke;
Pen.moveTo(50@210);
Pen.lineTo(50@215);
Pen.stroke;
Pen.moveTo(60@210);
Pen.lineTo(60@215);
Pen.stroke;
Pen.moveTo(70@210);
Pen.lineTo(70@215);
Pen.stroke;
Pen.moveTo(80@210);
Pen.lineTo(80@215);
Pen.stroke;
Pen.moveTo(90@210);
Pen.lineTo(90@215);
```

```
        Pen.stroke;
        Pen.moveTo(100@210);
        Pen.lineTo(100@215);
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@173.1);
Pen.lineTo(130@175.8);
Pen.lineTo(130@119.7);
Pen.lineTo(10@125);
        Pen.fill;

        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@126.7);
        Pen.lineTo(70@128);
        Pen.stroke;
        Pen.moveTo(70@200);
        Pen.lineTo(130@202.2);
        Pen.stroke;
        Pen.moveTo(10@146.5);
        Pen.lineTo(70@147.7);
        Pen.stroke;
        Pen.moveTo(10@167.7);
        Pen.lineTo(70@168.7);
        Pen.stroke;
        Pen.moveTo(10@193.8);
        Pen.lineTo(70@192.5);
        Pen.stroke;


        Pen.fillColor = Color.blue(1, 0.9);
Pen.moveTo(10@10);

Pen.lineTo(30@10);
Pen.lineTo(30@20);
Pen.lineTo(10@20);
Pen.lineTo(10@10);
        Pen.fill;

        Pen.fillColor = Color.red(1, 0.9);
Pen.moveTo(62.62@10);
Pen.lineTo(79.2@10);
Pen.lineTo(79.2@20);
```

```
    Pen.lineTo(62.62@20);
    Pen.lineTo(62.62@10);
            Pen.fill;

            Pen.moveTo(98.32@10);
    Pen.lineTo(124.9@10);
    Pen.lineTo(124.9@20);
    Pen.lineTo(98.32@20);
    Pen.lineTo(98.32@10);
            Pen.fill;

            Pen.fillColor = Color.gray(0.1, 0.4);
            Pen.font = Font("Birch Std", 45);
            Pen.stringAtPoint("10", (100@40));

// button 11

            Pen.translate(157,0);
            Pen.strokeColor = Color.black;

            Pen.moveTo(0@0);
            Pen.lineTo(140@0);
            Pen.lineTo(140@220);
            Pen.lineTo(0@220);
            Pen.lineTo(0@0);
            Pen.stroke;

            Pen.moveTo(10@10);
    Pen.lineTo(10@210);
            Pen.lineTo(130@210);
    Pen.stroke;

            Pen.moveTo(5@10);
            Pen.lineTo(10@10);
            Pen.stroke;
            Pen.moveTo(5@20);
            Pen.lineTo(10@20);
            Pen.stroke;
            Pen.moveTo(5@30);
            Pen.lineTo(10@30);
            Pen.stroke;
            Pen.moveTo(5@40);
            Pen.lineTo(10@40);
            Pen.stroke;
            Pen.moveTo(5@50);
            Pen.lineTo(10@50);
            Pen.stroke;
            Pen.moveTo(5@60);
            Pen.lineTo(10@60);
            Pen.stroke;
            Pen.moveTo(5@70);
            Pen.lineTo(10@70);
            Pen.stroke;
            Pen.moveTo(5@80);
            Pen.lineTo(10@80);
```

```
Pen.stroke;
Pen.moveTo(5@90);
Pen.lineTo(10@90);
Pen.stroke;
Pen.moveTo(5@100);
Pen.lineTo(10@100);
Pen.stroke;
Pen.moveTo(5@110);
Pen.lineTo(10@110);
Pen.stroke;
Pen.moveTo(5@120);
Pen.lineTo(10@120);
Pen.stroke;
Pen.moveTo(5@130);
Pen.lineTo(10@130);
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
Pen.stroke;
Pen.moveTo(30@210);
Pen.lineTo(30@215);
Pen.stroke;
Pen.moveTo(40@210);
Pen.lineTo(40@215);
Pen.stroke;
Pen.moveTo(50@210);
Pen.lineTo(50@215);
Pen.stroke;
Pen.moveTo(60@210);
Pen.lineTo(60@215);
Pen.stroke;
Pen.moveTo(70@210);
Pen.lineTo(70@215);
```

```
        Pen.stroke;
        Pen.moveTo(80@210);
        Pen.lineTo(80@215);
        Pen.stroke;
        Pen.moveTo(90@210);
        Pen.lineTo(90@215);
        Pen.stroke;
        Pen.moveTo(100@210);
        Pen.lineTo(100@215);
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@175.8);
        Pen.lineTo(30@178.2);
        Pen.lineTo(76@177.8);
        Pen.lineTo(104@185.1);
        Pen.lineTo(118@184.9);
        Pen.lineTo(126@177);
Pen.lineTo(130@177.5);
Pen.lineTo(130@114.5);
        Pen.lineTo(96.4@105.8);
        Pen.lineTo(80@101.9);
        Pen.lineTo(58@151.7);
Pen.lineTo(10@119.7);
        Pen.fill;

        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@160.2);
        Pen.lineTo(70@161.9);
        Pen.stroke;
        Pen.moveTo(70@186.3);
        Pen.lineTo(130@187.5);
        Pen.stroke;
        Pen.moveTo(10@181.2);
        Pen.lineTo(70@181.5);
        Pen.stroke;

        Pen.fillColor = Color.blue(1, 0.9);
Pen.moveTo(10@10);

Pen.lineTo(16@10);
Pen.lineTo(16@20);
Pen.lineTo(10@20);
```

```
    Pen.lineTo(10@10);
            Pen.fill;

            Pen.fillColor = Color.red(1, 0.9);
    Pen.moveTo(40.58@10);
    Pen.lineTo(61.14@10);
    Pen.lineTo(61.14@20);
    Pen.lineTo(40.58@20);
    Pen.lineTo(40.58@10);
            Pen.fill;

            Pen.moveTo(103.14@10);
    Pen.lineTo(126.24@10);
    Pen.lineTo(126.24@20);
    Pen.lineTo(103.14@20);
    Pen.lineTo(103.14@10);
            Pen.fill;

            Pen.fillColor = Color.gray(0.1, 0.4);
            Pen.font = Font("Birch Std", 45);
            Pen.stringAtPoint("11", (100@40));

//button 12

            Pen.translate(157,0);
            Pen.strokeColor = Color.black;

            Pen.moveTo(0@0);
            Pen.lineTo(140@0);
            Pen.lineTo(140@220);
            Pen.lineTo(0@220);
            Pen.lineTo(0@0);
            Pen.stroke;

            Pen.moveTo(10@10);
    Pen.lineTo(10@210);
            Pen.lineTo(130@210);
    Pen.stroke;

            Pen.moveTo(5@10);
            Pen.lineTo(10@10);
            Pen.stroke;
            Pen.moveTo(5@20);
            Pen.lineTo(10@20);
            Pen.stroke;
            Pen.moveTo(5@30);
            Pen.lineTo(10@30);
            Pen.stroke;
            Pen.moveTo(5@40);
            Pen.lineTo(10@40);
            Pen.stroke;
            Pen.moveTo(5@50);
            Pen.lineTo(10@50);
            Pen.stroke;
            Pen.moveTo(5@60);
```

```
Pen.lineTo(10@60);
Pen.stroke;
Pen.moveTo(5@70);
Pen.lineTo(10@70);
Pen.stroke;
Pen.moveTo(5@80);
Pen.lineTo(10@80);
Pen.stroke;
Pen.moveTo(5@90);
Pen.lineTo(10@90);
Pen.stroke;
Pen.moveTo(5@100);
Pen.lineTo(10@100);
Pen.stroke;
Pen.moveTo(5@110);
Pen.lineTo(10@110);
Pen.stroke;
Pen.moveTo(5@120);
Pen.lineTo(10@120);
Pen.stroke;
Pen.moveTo(5@130);
Pen.lineTo(10@130);
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
Pen.stroke;
Pen.moveTo(30@210);
Pen.lineTo(30@215);
Pen.stroke;
Pen.moveTo(40@210);
Pen.lineTo(40@215);
Pen.stroke;
Pen.moveTo(50@210);
```

```
        Pen.lineTo(50@215);
        Pen.stroke;
        Pen.moveTo(60@210);
        Pen.lineTo(60@215);
        Pen.stroke;
        Pen.moveTo(70@210);
        Pen.lineTo(70@215);
        Pen.stroke;
        Pen.moveTo(80@210);
        Pen.lineTo(80@215);
        Pen.stroke;
        Pen.moveTo(90@210);
        Pen.lineTo(90@215);
        Pen.stroke;
        Pen.moveTo(100@210);
        Pen.lineTo(100@215);
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@177.5);
        Pen.lineTo(54.6@155.7);
        Pen.lineTo(60@173.4);
        Pen.lineTo(71@162.2);
        Pen.lineTo(87.4@153.2);
Pen.lineTo(130@179.9);
Pen.lineTo(130@108.8);
        Pen.lineTo(104.4@87.4);
        Pen.lineTo(68@94.5);
        Pen.lineTo(26@32.3);
Pen.lineTo(10@114.4);
        Pen.fill;

        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@151.5);
        Pen.lineTo(70@152.4);
        Pen.stroke;
        Pen.moveTo(70@170.7);
        Pen.lineTo(130@172.8);
        Pen.stroke;
        Pen.moveTo(10@180.5);
        Pen.lineTo(70@180.9);
        Pen.stroke;
        Pen.moveTo(70@182.1);
```

```
            Pen.lineTo(130@182.6);
            Pen.stroke;



            Pen.fillColor = Color.blue(1, 0.9);
    Pen.moveTo(10@10);

    Pen.lineTo(30@10);
    Pen.lineTo(30@20);
    Pen.lineTo(10@20);
    Pen.lineTo(10@10);
            Pen.fill;

            Pen.fillColor = Color.red(1, 0.9);
    Pen.moveTo(30.7@10);
    Pen.lineTo(60.42@10);
    Pen.lineTo(60.42@20);
    Pen.lineTo(30.7@20);
    Pen.lineTo(30.7@10);
            Pen.fill;

            Pen.moveTo(96.24@10);
    Pen.lineTo(123.66@10);
    Pen.lineTo(123.66@20);
    Pen.lineTo(96.24@20);
    Pen.lineTo(96.24@10);
            Pen.fill;

            Pen.fillColor = Color.gray(0.1, 0.4);
            Pen.font = Font("Birch Std", 45);
            Pen.stringAtPoint("12", (100@40));

//button 13

            Pen.translate(157,0);
            Pen.strokeColor = Color.black;

            Pen.moveTo(0@0);
            Pen.lineTo(140@0);
            Pen.lineTo(140@220);
            Pen.lineTo(0@220);
            Pen.lineTo(0@0);
            Pen.stroke;

            Pen.moveTo(10@10);
    Pen.lineTo(10@210);
            Pen.lineTo(130@210);
    Pen.stroke;

            Pen.moveTo(5@10);
            Pen.lineTo(10@10);
            Pen.stroke;
            Pen.moveTo(5@20);
            Pen.lineTo(10@20);
```

```
Pen.stroke;
Pen.moveTo(5@30);
Pen.lineTo(10@30);
Pen.stroke;
Pen.moveTo(5@40);
Pen.lineTo(10@40);
Pen.stroke;
Pen.moveTo(5@50);
Pen.lineTo(10@50);
Pen.stroke;
Pen.moveTo(5@60);
Pen.lineTo(10@60);
Pen.stroke;
Pen.moveTo(5@70);
Pen.lineTo(10@70);
Pen.stroke;
Pen.moveTo(5@80);
Pen.lineTo(10@80);
Pen.stroke;
Pen.moveTo(5@90);
Pen.lineTo(10@90);
Pen.stroke;
Pen.moveTo(5@100);
Pen.lineTo(10@100);
Pen.stroke;
Pen.moveTo(5@110);
Pen.lineTo(10@110);
Pen.stroke;
Pen.moveTo(5@120);
Pen.lineTo(10@120);
Pen.stroke;
Pen.moveTo(5@130);
Pen.lineTo(10@130);
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;
```

```
        Pen.moveTo(20@210);
        Pen.lineTo(20@215);
        Pen.stroke;
        Pen.moveTo(30@210);
        Pen.lineTo(30@215);
        Pen.stroke;
        Pen.moveTo(40@210);
        Pen.lineTo(40@215);
        Pen.stroke;
        Pen.moveTo(50@210);
        Pen.lineTo(50@215);
        Pen.stroke;
        Pen.moveTo(60@210);
        Pen.lineTo(60@215);
        Pen.stroke;
        Pen.moveTo(70@210);
        Pen.lineTo(70@215);
        Pen.stroke;
        Pen.moveTo(80@210);
        Pen.lineTo(80@215);
        Pen.stroke;
        Pen.moveTo(90@210);
        Pen.lineTo(90@215);
        Pen.stroke;
        Pen.moveTo(100@210);
        Pen.lineTo(100@215);
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@179.9);
        Pen.lineTo(30.4@116);
        Pen.lineTo(50.4@156.6);
        Pen.lineTo(70.4@152.6);
        Pen.lineTo(90.4@110);
Pen.lineTo(130@181.8);
Pen.lineTo(130@103);
        Pen.lineTo(90.4@101.3);
        Pen.lineTo(70.4@152.3);
        Pen.lineTo(50.4@150.2);
        Pen.lineTo(30.4@107.3);
Pen.lineTo(10@108.8);
        Pen.fill;
```

```
		Pen.strokeColor = Color.magenta(0.3, 0.9);
		Pen.width = 3;
		Pen.moveTo(10@187.5);
		Pen.lineTo(70@188.7);
		Pen.stroke;
		Pen.moveTo(70@188.5);
		Pen.lineTo(130@190.1);
		Pen.stroke;
		Pen.moveTo(70@144.8);
		Pen.lineTo(130@145.3);
		Pen.stroke;
		Pen.moveTo(70@164);
		Pen.lineTo(130@164.9);
		Pen.stroke;
		Pen.moveTo(70@191.2);
		Pen.lineTo(130@191.7);
		Pen.stroke;

		Pen.fillColor = Color.blue(1, 0.9);
	Pen.moveTo(10@10);

	Pen.lineTo(16@10);
	Pen.lineTo(16@20);
	Pen.lineTo(10@20);
	Pen.lineTo(10@10);
		Pen.fill;

		Pen.fillColor = Color.red(1, 0.9);
	Pen.moveTo(42.28@10);
	Pen.lineTo(64.58@10);
	Pen.lineTo(64.58@20);
	Pen.lineTo(42.28@20);
	Pen.lineTo(42.28@10);
		Pen.fill;

		Pen.moveTo(104.28@10);
	Pen.lineTo(128.56@10);
	Pen.lineTo(128.56@20);
	Pen.lineTo(104.28@20);
	Pen.lineTo(104.28@10);
		Pen.fill;

		Pen.fillColor = Color.gray(0.1, 0.4);
		Pen.font = Font("Birch Std", 45);
		Pen.stringAtPoint("13", (100@40));

//button 14

		Pen.translate(157,0);
		Pen.strokeColor = Color.black;

		Pen.moveTo(0@0);
		Pen.lineTo(140@0);
		Pen.lineTo(140@220);
		Pen.lineTo(0@220);
```

```
        Pen.lineTo(0@0);
        Pen.stroke;

        Pen.moveTo(10@10);
Pen.lineTo(10@210);
        Pen.lineTo(130@210);
Pen.stroke;

        Pen.moveTo(5@10);
        Pen.lineTo(10@10);
        Pen.stroke;
        Pen.moveTo(5@20);
        Pen.lineTo(10@20);
        Pen.stroke;
        Pen.moveTo(5@30);
        Pen.lineTo(10@30);
        Pen.stroke;
        Pen.moveTo(5@40);
        Pen.lineTo(10@40);
        Pen.stroke;
        Pen.moveTo(5@50);
        Pen.lineTo(10@50);
        Pen.stroke;
        Pen.moveTo(5@60);
        Pen.lineTo(10@60);
        Pen.stroke;
        Pen.moveTo(5@70);
        Pen.lineTo(10@70);
        Pen.stroke;
        Pen.moveTo(5@80);
        Pen.lineTo(10@80);
        Pen.stroke;
        Pen.moveTo(5@90);
        Pen.lineTo(10@90);
        Pen.stroke;
        Pen.moveTo(5@100);
        Pen.lineTo(10@100);
        Pen.stroke;
        Pen.moveTo(5@110);
        Pen.lineTo(10@110);
        Pen.stroke;
        Pen.moveTo(5@120);
        Pen.lineTo(10@120);
        Pen.stroke;
        Pen.moveTo(5@130);
        Pen.lineTo(10@130);
        Pen.stroke;
        Pen.moveTo(5@140);
        Pen.lineTo(10@140);
        Pen.stroke;
        Pen.moveTo(5@150);
        Pen.lineTo(10@150);
        Pen.stroke;
        Pen.moveTo(5@160);
        Pen.lineTo(10@160);
```

```
        Pen.stroke;
        Pen.moveTo(5@170);
        Pen.lineTo(10@170);
        Pen.stroke;
        Pen.moveTo(5@180);
        Pen.lineTo(10@180);
        Pen.stroke;
        Pen.moveTo(5@190);
        Pen.lineTo(10@190);
        Pen.stroke;
        Pen.moveTo(5@200);
        Pen.lineTo(10@200);
        Pen.stroke;

        Pen.moveTo(20@210);
        Pen.lineTo(20@215);
        Pen.stroke;
        Pen.moveTo(30@210);
        Pen.lineTo(30@215);
        Pen.stroke;
        Pen.moveTo(40@210);
        Pen.lineTo(40@215);
        Pen.stroke;
        Pen.moveTo(50@210);
        Pen.lineTo(50@215);
        Pen.stroke;
        Pen.moveTo(60@210);
        Pen.lineTo(60@215);
        Pen.stroke;
        Pen.moveTo(70@210);
        Pen.lineTo(70@215);
        Pen.stroke;
        Pen.moveTo(80@210);
        Pen.lineTo(80@215);
        Pen.stroke;
        Pen.moveTo(90@210);
        Pen.lineTo(90@215);
        Pen.stroke;
        Pen.moveTo(100@210);
        Pen.lineTo(100@215);
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@181.8);
```

```
    Pen.lineTo(130@184.5);
    Pen.lineTo(130@97.2);
            Pen.lineTo(114.4@104.2);
            Pen.lineTo(100.4@148);
            Pen.lineTo(74.4@80.4);
            Pen.lineTo(64.4@62.4);
            Pen.lineTo(40.4@128.4);
            Pen.lineTo(24.4@134.2);
    Pen.lineTo(10@103);
            Pen.fill;

            Pen.strokeColor = Color.magenta(0.3, 0.9);
            Pen.width = 3;
            Pen.moveTo(10@191.1);
            Pen.lineTo(70@192.3);
            Pen.stroke;
            Pen.moveTo(70@185.5);
            Pen.lineTo(130@186.2);
            Pen.stroke;


            Pen.fillColor = Color.blue(1, 0.9);
    Pen.moveTo(10@10);

    Pen.lineTo(30@10);
    Pen.lineTo(30@20);
    Pen.lineTo(10@20);
    Pen.lineTo(10@10);
            Pen.fill;

            Pen.fillColor = Color.red(1, 0.9);
    Pen.moveTo(53.6@10);
    Pen.lineTo(73.88@10);
    Pen.lineTo(73.88@20);
    Pen.lineTo(53.6@20);
    Pen.lineTo(53.66@10);
            Pen.fill;

            Pen.moveTo(109.42@10);
    Pen.lineTo(131.42@10);
    Pen.lineTo(131.42@20);
    Pen.lineTo(109.42@20);
    Pen.lineTo(109.42@10);
            Pen.fill;

            Pen.fillColor = Color.gray(0.1, 0.4);
            Pen.font = Font("Birch Std", 45);
            Pen.stringAtPoint("14", (100@40));

//button 15

            Pen.translate(157,0);
            Pen.strokeColor = Color.black;

            Pen.moveTo(0@0);
```

```
        Pen.lineTo(140@0);
        Pen.lineTo(140@220);
        Pen.lineTo(0@220);
        Pen.lineTo(0@0);
        Pen.stroke;

        Pen.moveTo(10@10);
Pen.lineTo(10@210);
        Pen.lineTo(130@210);
Pen.stroke;

        Pen.moveTo(5@10);
        Pen.lineTo(10@10);
        Pen.stroke;
        Pen.moveTo(5@20);
        Pen.lineTo(10@20);
        Pen.stroke;
        Pen.moveTo(5@30);
        Pen.lineTo(10@30);
        Pen.stroke;
        Pen.moveTo(5@40);
        Pen.lineTo(10@40);
        Pen.stroke;
        Pen.moveTo(5@50);
        Pen.lineTo(10@50);
        Pen.stroke;
        Pen.moveTo(5@60);
        Pen.lineTo(10@60);
        Pen.stroke;
        Pen.moveTo(5@70);
        Pen.lineTo(10@70);
        Pen.stroke;
        Pen.moveTo(5@80);
        Pen.lineTo(10@80);
        Pen.stroke;
        Pen.moveTo(5@90);
        Pen.lineTo(10@90);
        Pen.stroke;
        Pen.moveTo(5@100);
        Pen.lineTo(10@100);
        Pen.stroke;
        Pen.moveTo(5@110);
        Pen.lineTo(10@110);
        Pen.stroke;
        Pen.moveTo(5@120);
        Pen.lineTo(10@120);
        Pen.stroke;
        Pen.moveTo(5@130);
        Pen.lineTo(10@130);
        Pen.stroke;
        Pen.moveTo(5@140);
        Pen.lineTo(10@140);
        Pen.stroke;
        Pen.moveTo(5@150);
        Pen.lineTo(10@150);
```

```
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
Pen.stroke;
Pen.moveTo(30@210);
Pen.lineTo(30@215);
Pen.stroke;
Pen.moveTo(40@210);
Pen.lineTo(40@215);
Pen.stroke;
Pen.moveTo(50@210);
Pen.lineTo(50@215);
Pen.stroke;
Pen.moveTo(60@210);
Pen.lineTo(60@215);
Pen.stroke;
Pen.moveTo(70@210);
Pen.lineTo(70@215);
Pen.stroke;
Pen.moveTo(80@210);
Pen.lineTo(80@215);
Pen.stroke;
Pen.moveTo(90@210);
Pen.lineTo(90@215);
Pen.stroke;
Pen.moveTo(100@210);
Pen.lineTo(100@215);
Pen.stroke;
Pen.moveTo(110@210);
Pen.lineTo(110@215);
Pen.stroke;
Pen.moveTo(120@210);
Pen.lineTo(120@215);
Pen.stroke;
Pen.moveTo(130@210);
Pen.lineTo(130@215);
Pen.stroke;

Pen.fillColor = Color.green(0.5, 0.9);
```

```
Pen.moveTo(10@107);

Pen.lineTo(10@184.5);
        Pen.lineTo(26@177.8);
        Pen.lineTo(44.4@181.8);
        Pen.lineTo(68@181);
        Pen.lineTo(108.4@169.1);
        Pen.lineTo(114.4@163.7);
        Pen.lineTo(124.4@185.8);
Pen.lineTo(130@186.5);
Pen.lineTo(130@91.2);
Pen.lineTo(10@97.2);
        Pen.fill;

        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@136.8);
        Pen.lineTo(70@138.2);
        Pen.stroke;
        Pen.moveTo(70@202.9);
        Pen.lineTo(130@210);
        Pen.stroke;
        Pen.moveTo(10@148.8);
        Pen.lineTo(70@149.5);
        Pen.stroke;
        Pen.moveTo(10@173.2);
        Pen.lineTo(70@174.1);
        Pen.stroke;
        Pen.moveTo(10@185);
        Pen.lineTo(70@185.7);
        Pen.stroke;


        Pen.fillColor = Color.blue(1, 0.9);
Pen.moveTo(10@10);

Pen.lineTo(16@10);
Pen.lineTo(16@20);
Pen.lineTo(10@20);
Pen.lineTo(10@10);
        Pen.fill;

        Pen.fillColor = Color.red(1, 0.9);
Pen.moveTo(41.42@10);
Pen.lineTo(62.84@10);
Pen.lineTo(62.84@20);
Pen.lineTo(41.42@20);
Pen.lineTo(41.42@10);
        Pen.fill;

        Pen.moveTo(96.86@10);
Pen.lineTo(113.72@10);
Pen.lineTo(113.72@20);
Pen.lineTo(96.86@20);
Pen.lineTo(96.86@10);
```

```
        Pen.fill;

        Pen.fillColor = Color.gray(0.1, 0.4);
        Pen.font = Font("Birch Std", 45);
        Pen.stringAtPoint("15", (100@40));

// button 16

        Pen.translate(157,0);
        Pen.strokeColor = Color.black;

        Pen.moveTo(0@0);
        Pen.lineTo(140@0);
        Pen.lineTo(140@220);
        Pen.lineTo(0@220);
        Pen.lineTo(0@0);
        Pen.stroke;

        Pen.moveTo(10@10);
    Pen.lineTo(10@210);
        Pen.lineTo(130@210);
    Pen.stroke;

        Pen.moveTo(5@10);
        Pen.lineTo(10@10);
        Pen.stroke;
        Pen.moveTo(5@20);
        Pen.lineTo(10@20);
        Pen.stroke;
        Pen.moveTo(5@30);
        Pen.lineTo(10@30);
        Pen.stroke;
        Pen.moveTo(5@40);
        Pen.lineTo(10@40);
        Pen.stroke;
        Pen.moveTo(5@50);
        Pen.lineTo(10@50);
        Pen.stroke;
        Pen.moveTo(5@60);
        Pen.lineTo(10@60);
        Pen.stroke;
        Pen.moveTo(5@70);
        Pen.lineTo(10@70);
        Pen.stroke;
        Pen.moveTo(5@80);
        Pen.lineTo(10@80);
        Pen.stroke;
        Pen.moveTo(5@90);
        Pen.lineTo(10@90);
        Pen.stroke;
        Pen.moveTo(5@100);
        Pen.lineTo(10@100);
        Pen.stroke;
        Pen.moveTo(5@110);
        Pen.lineTo(10@110);
```

```
Pen.stroke;
Pen.moveTo(5@120);
Pen.lineTo(10@120);
Pen.stroke;
Pen.moveTo(5@130);
Pen.lineTo(10@130);
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
Pen.stroke;
Pen.moveTo(30@210);
Pen.lineTo(30@215);
Pen.stroke;
Pen.moveTo(40@210);
Pen.lineTo(40@215);
Pen.stroke;
Pen.moveTo(50@210);
Pen.lineTo(50@215);
Pen.stroke;
Pen.moveTo(60@210);
Pen.lineTo(60@215);
Pen.stroke;
Pen.moveTo(70@210);
Pen.lineTo(70@215);
Pen.stroke;
Pen.moveTo(80@210);
Pen.lineTo(80@215);
Pen.stroke;
Pen.moveTo(90@210);
Pen.lineTo(90@215);
Pen.stroke;
Pen.moveTo(100@210);
Pen.lineTo(100@215);
```

```
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@186.5);
        Pen.lineTo(22.4@110.4);
        Pen.lineTo(56.4@108.8);
        Pen.lineTo(82.4@180.8);
        Pen.lineTo(92.4@176.1);
        Pen.lineTo(100.4@196.1);
        Pen.lineTo(122.4@191.5);
Pen.lineTo(130@189);
Pen.lineTo(130@85);
        Pen.lineTo(104.4@74.1);
        Pen.lineTo(102.2@137.9);
        Pen.lineTo(90.4@116.4);
        Pen.lineTo(86.4@101.5);
        Pen.lineTo(32@101.9);
Pen.lineTo(10@91.2);
        Pen.fill;

        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@154.2);
        Pen.lineTo(70@155.3);
        Pen.stroke;
        Pen.moveTo(70@177.2);
        Pen.lineTo(130@179.2);
        Pen.stroke;
        Pen.moveTo(10@180.5);
        Pen.lineTo(70@180.9);
        Pen.stroke;
        Pen.moveTo(70@182.2);
        Pen.lineTo(130@182.7);
        Pen.stroke;

        Pen.fillColor = Color.blue(1, 0.9);
Pen.moveTo(10@10);

Pen.lineTo(30@10);
Pen.lineTo(30@20);
Pen.lineTo(10@20);
Pen.lineTo(10@10);
        Pen.fill;
```

```
		Pen.fillColor = Color.red(1, 0.9);
	Pen.moveTo(63.3@10);
	Pen.lineTo(79.58@10);
	Pen.lineTo(79.58@20);
	Pen.lineTo(63.3@20);
	Pen.lineTo(63.3@10);
		Pen.fill;

		Pen.moveTo(105.26@10);
	Pen.lineTo(128.98@10);
	Pen.lineTo(128.98@20);
	Pen.lineTo(105.26@20);
	Pen.lineTo(105.26@10);
		Pen.fill;

		Pen.fillColor = Color.gray(0.1, 0.4);
		Pen.font = Font("Birch Std", 45);
		Pen.stringAtPoint("16", (100@40));

// button 17

		Pen.translate(-1256,240);
		Pen.strokeColor = Color.black;

		Pen.moveTo(0@0);
		Pen.lineTo(140@0);
		Pen.lineTo(140@220);
		Pen.lineTo(0@220);
		Pen.lineTo(0@0);
		Pen.stroke;

		Pen.moveTo(10@10);
	Pen.lineTo(10@210);
		Pen.lineTo(130@210);
	Pen.stroke;

		Pen.moveTo(5@10);
		Pen.lineTo(10@10);
		Pen.stroke;
		Pen.moveTo(5@20);
		Pen.lineTo(10@20);
		Pen.stroke;
		Pen.moveTo(5@30);
		Pen.lineTo(10@30);
		Pen.stroke;
		Pen.moveTo(5@40);
		Pen.lineTo(10@40);
		Pen.stroke;
		Pen.moveTo(5@50);
		Pen.lineTo(10@50);
		Pen.stroke;
		Pen.moveTo(5@60);
		Pen.lineTo(10@60);
		Pen.stroke;
		Pen.moveTo(5@70);
```

```
Pen.lineTo(10@70);
Pen.stroke;
Pen.moveTo(5@80);
Pen.lineTo(10@80);
Pen.stroke;
Pen.moveTo(5@90);
Pen.lineTo(10@90);
Pen.stroke;
Pen.moveTo(5@100);
Pen.lineTo(10@100);
Pen.stroke;
Pen.moveTo(5@110);
Pen.lineTo(10@110);
Pen.stroke;
Pen.moveTo(5@120);
Pen.lineTo(10@120);
Pen.stroke;
Pen.moveTo(5@130);
Pen.lineTo(10@130);
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
Pen.stroke;
Pen.moveTo(30@210);
Pen.lineTo(30@215);
Pen.stroke;
Pen.moveTo(40@210);
Pen.lineTo(40@215);
Pen.stroke;
Pen.moveTo(50@210);
Pen.lineTo(50@215);
Pen.stroke;
Pen.moveTo(60@210);
```

```
        Pen.lineTo(60@215);
        Pen.stroke;
        Pen.moveTo(70@210);
        Pen.lineTo(70@215);
        Pen.stroke;
        Pen.moveTo(80@210);
        Pen.lineTo(80@215);
        Pen.stroke;
        Pen.moveTo(90@210);
        Pen.lineTo(90@215);
        Pen.stroke;
        Pen.moveTo(100@210);
        Pen.lineTo(100@215);
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@189);
Pen.lineTo(130@191.2);
Pen.lineTo(130@78.7);
Pen.lineTo(10@85);
        Pen.fill;

        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@120.2);
        Pen.lineTo(70@121.6);
        Pen.stroke;
        Pen.moveTo(70@175);
        Pen.lineTo(130@177.3);
        Pen.stroke;
        Pen.moveTo(10@145.2);
        Pen.lineTo(70@145.8);
        Pen.stroke;
        Pen.moveTo(10@164.7);
        Pen.lineTo(70@165.5);
        Pen.stroke;
        Pen.moveTo(70@182.2);
        Pen.lineTo(130@182.6);
        Pen.stroke;
        Pen.moveTo(10@201.1);
        Pen.lineTo(70@203.3);
        Pen.stroke;
```

```
		Pen.fillColor = Color.blue(1, 0.9);
	Pen.moveTo(10@10);

	Pen.lineTo(16@10);
	Pen.lineTo(16@20);
	Pen.lineTo(10@20);
	Pen.lineTo(10@10);
		Pen.fill;

		Pen.fillColor = Color.red(1, 0.9);
	Pen.moveTo(42@10);
	Pen.lineTo(64@10);
	Pen.lineTo(64@20);
	Pen.lineTo(42@20);
	Pen.lineTo(42@10);
		Pen.fill;

		Pen.moveTo(107.72@10);
	Pen.lineTo(135.42@10);
	Pen.lineTo(135.42@20);
	Pen.lineTo(107.72@20);
	Pen.lineTo(107.72@10);
		Pen.fill;

		Pen.fillColor = Color.gray(0.1, 0.4);
		Pen.font = Font("Birch Std", 45);
		Pen.stringAtPoint("17", (100@40));

//button 18

		Pen.translate(157,0);
		Pen.strokeColor = Color.black;

		Pen.moveTo(0@0);
		Pen.lineTo(140@0);
		Pen.lineTo(140@220);
		Pen.lineTo(0@220);
		Pen.lineTo(0@0);
		Pen.stroke;

		Pen.moveTo(10@10);
	Pen.lineTo(10@210);
		Pen.lineTo(130@210);
	Pen.stroke;

		Pen.moveTo(5@10);
		Pen.lineTo(10@10);
		Pen.stroke;
		Pen.moveTo(5@20);
		Pen.lineTo(10@20);
		Pen.stroke;
		Pen.moveTo(5@30);
		Pen.lineTo(10@30);
		Pen.stroke;
		Pen.moveTo(5@40);
```

```
Pen.lineTo(10@40);
Pen.stroke;
Pen.moveTo(5@50);
Pen.lineTo(10@50);
Pen.stroke;
Pen.moveTo(5@60);
Pen.lineTo(10@60);
Pen.stroke;
Pen.moveTo(5@70);
Pen.lineTo(10@70);
Pen.stroke;
Pen.moveTo(5@80);
Pen.lineTo(10@80);
Pen.stroke;
Pen.moveTo(5@90);
Pen.lineTo(10@90);
Pen.stroke;
Pen.moveTo(5@100);
Pen.lineTo(10@100);
Pen.stroke;
Pen.moveTo(5@110);
Pen.lineTo(10@110);
Pen.stroke;
Pen.moveTo(5@120);
Pen.lineTo(10@120);
Pen.stroke;
Pen.moveTo(5@130);
Pen.lineTo(10@130);
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
Pen.stroke;
Pen.moveTo(30@210);
```

```
        Pen.lineTo(30@215);
        Pen.stroke;
        Pen.moveTo(40@210);
        Pen.lineTo(40@215);
        Pen.stroke;
        Pen.moveTo(50@210);
        Pen.lineTo(50@215);
        Pen.stroke;
        Pen.moveTo(60@210);
        Pen.lineTo(60@215);
        Pen.stroke;
        Pen.moveTo(70@210);
        Pen.lineTo(70@215);
        Pen.stroke;
        Pen.moveTo(80@210);
        Pen.lineTo(80@215);
        Pen.stroke;
        Pen.moveTo(90@210);
        Pen.lineTo(90@215);
        Pen.stroke;
        Pen.moveTo(100@210);
        Pen.lineTo(100@215);
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@191.2);
        Pen.lineTo(26.4@101.9);
        Pen.lineTo(44.4@152.2);
        Pen.lineTo(62.4@184.9);
        Pen.lineTo(88.4@110);
        Pen.lineTo(100.4@180.8);
        Pen.lineTo(106.4@156.7);
        Pen.lineTo(120.4@118.6);
Pen.lineTo(130@193.6);
Pen.lineTo(130@72.5);
        Pen.lineTo(120.4@94.5);
        Pen.lineTo(106.4@149.1);
        Pen.lineTo(100.4@150.5);
        Pen.lineTo(88.4@101.6);
        Pen.lineTo(62.4@101.9);
        Pen.lineTo(44.4@116.8);
        Pen.lineTo(26.4@92.2);
Pen.lineTo(10@78.7);
        Pen.fill;
```

```
        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@128.1);
        Pen.lineTo(70@130.1);
        Pen.stroke;
        Pen.moveTo(70@152.8);
        Pen.lineTo(130@154.3);
        Pen.stroke;
        Pen.moveTo(10@147.1);
        Pen.lineTo(70@147.6);
        Pen.stroke;
        Pen.moveTo(10@168.6);
        Pen.lineTo(70@169.3);
        Pen.stroke;
        Pen.moveTo(70@180.6);
        Pen.lineTo(130@181);
        Pen.stroke;
        Pen.moveTo(10@188.2);
        Pen.lineTo(70@190);
        Pen.stroke;


        Pen.fillColor = Color.blue(1, 0.9);
    Pen.moveTo(10@10);

    Pen.lineTo(30@10);
    Pen.lineTo(30@20);
    Pen.lineTo(10@20);
    Pen.lineTo(10@10);
        Pen.fill;

        Pen.fillColor = Color.red(1, 0.9);
    Pen.moveTo(42.48@10);
    Pen.lineTo(67.34@10);
    Pen.lineTo(67.34@20);
    Pen.lineTo(42.48@20);
    Pen.lineTo(42.48@10);
        Pen.fill;

        Pen.moveTo(96.94@10);
    Pen.lineTo(124.08@10);
    Pen.lineTo(124.08@20);
    Pen.lineTo(96.94@20);
    Pen.lineTo(96.94@10);
        Pen.fill;

        Pen.fillColor = Color.gray(0.1, 0.4);
        Pen.font = Font("Birch Std", 45);
        Pen.stringAtPoint("18", (100@40));


//button 19

        Pen.translate(157,0);
```

```
        Pen.strokeColor = Color.black;

        Pen.moveTo(0@0);
        Pen.lineTo(140@0);
        Pen.lineTo(140@220);
        Pen.lineTo(0@220);
        Pen.lineTo(0@0);
        Pen.stroke;

        Pen.moveTo(10@10);
Pen.lineTo(10@210);
        Pen.lineTo(130@210);
Pen.stroke;

        Pen.moveTo(5@10);
        Pen.lineTo(10@10);
        Pen.stroke;
        Pen.moveTo(5@20);
        Pen.lineTo(10@20);
        Pen.stroke;
        Pen.moveTo(5@30);
        Pen.lineTo(10@30);
        Pen.stroke;
        Pen.moveTo(5@40);
        Pen.lineTo(10@40);
        Pen.stroke;
        Pen.moveTo(5@50);
        Pen.lineTo(10@50);
        Pen.stroke;
        Pen.moveTo(5@60);
        Pen.lineTo(10@60);
        Pen.stroke;
        Pen.moveTo(5@70);
        Pen.lineTo(10@70);
        Pen.stroke;
        Pen.moveTo(5@80);
        Pen.lineTo(10@80);
        Pen.stroke;
        Pen.moveTo(5@90);
        Pen.lineTo(10@90);
        Pen.stroke;
        Pen.moveTo(5@100);
        Pen.lineTo(10@100);
        Pen.stroke;
        Pen.moveTo(5@110);
        Pen.lineTo(10@110);
        Pen.stroke;
        Pen.moveTo(5@120);
        Pen.lineTo(10@120);
        Pen.stroke;
        Pen.moveTo(5@130);
        Pen.lineTo(10@130);
        Pen.stroke;
        Pen.moveTo(5@140);
        Pen.lineTo(10@140);
```

```
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
Pen.stroke;
Pen.moveTo(30@210);
Pen.lineTo(30@215);
Pen.stroke;
Pen.moveTo(40@210);
Pen.lineTo(40@215);
Pen.stroke;
Pen.moveTo(50@210);
Pen.lineTo(50@215);
Pen.stroke;
Pen.moveTo(60@210);
Pen.lineTo(60@215);
Pen.stroke;
Pen.moveTo(70@210);
Pen.lineTo(70@215);
Pen.stroke;
Pen.moveTo(80@210);
Pen.lineTo(80@215);
Pen.stroke;
Pen.moveTo(90@210);
Pen.lineTo(90@215);
Pen.stroke;
Pen.moveTo(100@210);
Pen.lineTo(100@215);
Pen.stroke;
Pen.moveTo(110@210);
Pen.lineTo(110@215);
Pen.stroke;
Pen.moveTo(120@210);
Pen.lineTo(120@215);
Pen.stroke;
Pen.moveTo(130@210);
Pen.lineTo(130@215);
```

```
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@193.6);
Pen.lineTo(130@195.5);
Pen.lineTo(130@63.2);
        Pen.lineTo(110.4@178.2);
        Pen.lineTo(70.4@174.5);
        Pen.lineTo(30.4@165.7);
Pen.lineTo(10@72.5);
        Pen.fill;

        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@195.8);
        Pen.lineTo(70@197.5);
        Pen.stroke;
        Pen.moveTo(70@138.7);
        Pen.lineTo(130@139.4);
        Pen.stroke;
        Pen.moveTo(70@149.6);
        Pen.lineTo(130@149.9);
        Pen.stroke;
        Pen.moveTo(70@174.1);
        Pen.lineTo(130@175.3);
        Pen.stroke;
        Pen.moveTo(70@185.7);
        Pen.lineTo(130@188.1);
        Pen.stroke;


        Pen.fillColor = Color.blue(1, 0.9);
Pen.moveTo(10@10);

Pen.lineTo(16@10);
Pen.lineTo(16@20);
Pen.lineTo(10@20);
Pen.lineTo(10@10);
        Pen.fill;

        Pen.fillColor = Color.red(1, 0.9);
Pen.moveTo(40@10);
Pen.lineTo(60@10);
Pen.lineTo(60@20);
Pen.lineTo(40@20);
Pen.lineTo(40@10);
        Pen.fill;

        Pen.moveTo(98.58@10);
Pen.lineTo(117.16@10);
Pen.lineTo(117.16@20);
Pen.lineTo(98.58@20);
Pen.lineTo(98.58@10);
```

```
        Pen.fill;

        Pen.fillColor = Color.gray(0.1, 0.4);
        Pen.font = Font("Birch Std", 45);
        Pen.stringAtPoint("19", (100@40));

        //button 20

        Pen.translate(157,0);
        Pen.strokeColor = Color.black;

        Pen.moveTo(0@0);
        Pen.lineTo(140@0);
        Pen.lineTo(140@220);
        Pen.lineTo(0@220);
        Pen.lineTo(0@0);
        Pen.stroke;

        Pen.moveTo(10@10);
Pen.lineTo(10@210);
        Pen.lineTo(130@210);
Pen.stroke;

        Pen.moveTo(5@10);
        Pen.lineTo(10@10);
        Pen.stroke;
        Pen.moveTo(5@20);
        Pen.lineTo(10@20);
        Pen.stroke;
        Pen.moveTo(5@30);
        Pen.lineTo(10@30);
        Pen.stroke;
        Pen.moveTo(5@40);
        Pen.lineTo(10@40);
        Pen.stroke;
        Pen.moveTo(5@50);
        Pen.lineTo(10@50);
        Pen.stroke;
        Pen.moveTo(5@60);
        Pen.lineTo(10@60);
        Pen.stroke;
        Pen.moveTo(5@70);
        Pen.lineTo(10@70);
        Pen.stroke;
        Pen.moveTo(5@80);
        Pen.lineTo(10@80);
        Pen.stroke;
        Pen.moveTo(5@90);
        Pen.lineTo(10@90);
        Pen.stroke;
        Pen.moveTo(5@100);
        Pen.lineTo(10@100);
        Pen.stroke;
        Pen.moveTo(5@110);
        Pen.lineTo(10@110);
```

```
Pen.stroke;
Pen.moveTo(5@120);
Pen.lineTo(10@120);
Pen.stroke;
Pen.moveTo(5@130);
Pen.lineTo(10@130);
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
Pen.stroke;
Pen.moveTo(30@210);
Pen.lineTo(30@215);
Pen.stroke;
Pen.moveTo(40@210);
Pen.lineTo(40@215);
Pen.stroke;
Pen.moveTo(50@210);
Pen.lineTo(50@215);
Pen.stroke;
Pen.moveTo(60@210);
Pen.lineTo(60@215);
Pen.stroke;
Pen.moveTo(70@210);
Pen.lineTo(70@215);
Pen.stroke;
Pen.moveTo(80@210);
Pen.lineTo(80@215);
Pen.stroke;
Pen.moveTo(90@210);
Pen.lineTo(90@215);
Pen.stroke;
Pen.moveTo(100@210);
Pen.lineTo(100@215);
```

```
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@195.5);
        Pen.lineTo(30.4@94.8);
        Pen.lineTo(76.4@71.2);
        Pen.lineTo(96.4@73.7);
Pen.lineTo(130@197.9);
Pen.lineTo(130@56.5);
Pen.lineTo(10@63.2);
        Pen.fill;

        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@158.3);
        Pen.lineTo(70@160.2);
        Pen.stroke;
        Pen.moveTo(70@184.4);
        Pen.lineTo(130@185.3);
        Pen.stroke;
        Pen.moveTo(10@181.1);
        Pen.lineTo(70@181.8);
        Pen.stroke;

        Pen.fillColor = Color.blue(1, 0.9);
Pen.moveTo(10@10);

Pen.lineTo(30@10);
Pen.lineTo(30@20);
Pen.lineTo(10@20);
Pen.lineTo(10@10);
        Pen.fill;

        Pen.fillColor = Color.red(1, 0.9);
Pen.moveTo(34.16@10);
Pen.lineTo(62.44@10);
Pen.lineTo(62.44@20);
Pen.lineTo(34.16@20);
Pen.lineTo(34.16@10);
        Pen.fill;

        Pen.moveTo(112.9@10);
Pen.lineTo(133.58@10);
Pen.lineTo(133.58@20);
```

```
    Pen.lineTo(112.9@20);
    Pen.lineTo(112.9@10);
            Pen.fill;

            Pen.fillColor = Color.gray(0.1, 0.4);
            Pen.font = Font("Birch Std", 45);
            Pen.stringAtPoint("20", (100@40));

//button 21

            Pen.translate(157,0);
            Pen.strokeColor = Color.black;

            Pen.moveTo(0@0);
            Pen.lineTo(140@0);
            Pen.lineTo(140@220);
            Pen.lineTo(0@220);
            Pen.lineTo(0@0);
            Pen.stroke;

            Pen.moveTo(10@10);
    Pen.lineTo(10@210);
            Pen.lineTo(130@210);
    Pen.stroke;

            Pen.moveTo(5@10);
            Pen.lineTo(10@10);
            Pen.stroke;
            Pen.moveTo(5@20);
            Pen.lineTo(10@20);
            Pen.stroke;
            Pen.moveTo(5@30);
            Pen.lineTo(10@30);
            Pen.stroke;
            Pen.moveTo(5@40);
            Pen.lineTo(10@40);
            Pen.stroke;
            Pen.moveTo(5@50);
            Pen.lineTo(10@50);
            Pen.stroke;
            Pen.moveTo(5@60);
            Pen.lineTo(10@60);
            Pen.stroke;
            Pen.moveTo(5@70);
            Pen.lineTo(10@70);
            Pen.stroke;
            Pen.moveTo(5@80);
            Pen.lineTo(10@80);
            Pen.stroke;
            Pen.moveTo(5@90);
            Pen.lineTo(10@90);
            Pen.stroke;
            Pen.moveTo(5@100);
            Pen.lineTo(10@100);
            Pen.stroke;
```

```
Pen.moveTo(5@110);
Pen.lineTo(10@110);
Pen.stroke;
Pen.moveTo(5@120);
Pen.lineTo(10@120);
Pen.stroke;
Pen.moveTo(5@130);
Pen.lineTo(10@130);
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
Pen.stroke;
Pen.moveTo(30@210);
Pen.lineTo(30@215);
Pen.stroke;
Pen.moveTo(40@210);
Pen.lineTo(40@215);
Pen.stroke;
Pen.moveTo(50@210);
Pen.lineTo(50@215);
Pen.stroke;
Pen.moveTo(60@210);
Pen.lineTo(60@215);
Pen.stroke;
Pen.moveTo(70@210);
Pen.lineTo(70@215);
Pen.stroke;
Pen.moveTo(80@210);
Pen.lineTo(80@215);
Pen.stroke;
Pen.moveTo(90@210);
Pen.lineTo(90@215);
Pen.stroke;
```

```
        Pen.moveTo(100@210);
        Pen.lineTo(100@215);
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@197.9);
        Pen.lineTo(20.3@180.8);
        Pen.lineTo(60.4@193.2);
        Pen.lineTo(70.4@200);
        Pen.lineTo(80.4@179);
Pen.lineTo(130@200);
Pen.lineTo(130@50.2);
        Pen.lineTo(100.4@41.3);
        Pen.lineTo(74.4@122.5);
        Pen.lineTo(44.4@178.5);
        Pen.lineTo(30.4@68.2);
Pen.lineTo(10@56.5);
        Pen.fill;

        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@168.8);
        Pen.lineTo(70@169.6);
        Pen.stroke;
        Pen.moveTo(70@124.6);
        Pen.lineTo(130@125.4);
        Pen.stroke;
        Pen.moveTo(70@146.3);
        Pen.lineTo(130@146.8);
        Pen.stroke;
        Pen.moveTo(10@181.8);
        Pen.lineTo(70@182.3);
        Pen.stroke;
        Pen.moveTo(70@167.2);
        Pen.lineTo(130@168.1);
        Pen.stroke;
        Pen.moveTo(70@195.2);
        Pen.lineTo(130@195.9);
        Pen.stroke;

        Pen.fillColor = Color.blue(1, 0.9);
Pen.moveTo(10@10);

Pen.lineTo(16@10);
```

```
    Pen.lineTo(16@20);
    Pen.lineTo(10@20);
    Pen.lineTo(10@10);
            Pen.fill;

            Pen.fillColor = Color.red(1, 0.9);
    Pen.moveTo(49.42@10);
    Pen.lineTo(78.84@10);
    Pen.lineTo(78.84@20);
    Pen.lineTo(49.42@20);
    Pen.lineTo(49.42@10);
            Pen.fill;

            Pen.moveTo(98.86@10);
    Pen.lineTo(117.72@10);
    Pen.lineTo(117.72@20);
    Pen.lineTo(98.86@20);
    Pen.lineTo(98.86@10);
            Pen.fill;

            Pen.fillColor = Color.gray(0.1, 0.4);
            Pen.font = Font("Birch Std", 45);
            Pen.stringAtPoint("21", (100@40));

//button 22

            Pen.translate(157,0);
            Pen.strokeColor = Color.black;

            Pen.moveTo(0@0);
            Pen.lineTo(140@0);
            Pen.lineTo(140@220);
            Pen.lineTo(0@220);
            Pen.lineTo(0@0);
            Pen.stroke;

            Pen.moveTo(10@10);
    Pen.lineTo(10@210);
            Pen.lineTo(130@210);
    Pen.stroke;

            Pen.moveTo(5@10);
            Pen.lineTo(10@10);
            Pen.stroke;
            Pen.moveTo(5@20);
            Pen.lineTo(10@20);
            Pen.stroke;
            Pen.moveTo(5@30);
            Pen.lineTo(10@30);
            Pen.stroke;
            Pen.moveTo(5@40);
            Pen.lineTo(10@40);
            Pen.stroke;
            Pen.moveTo(5@50);
            Pen.lineTo(10@50);
```

```
Pen.stroke;
Pen.moveTo(5@60);
Pen.lineTo(10@60);
Pen.stroke;
Pen.moveTo(5@70);
Pen.lineTo(10@70);
Pen.stroke;
Pen.moveTo(5@80);
Pen.lineTo(10@80);
Pen.stroke;
Pen.moveTo(5@90);
Pen.lineTo(10@90);
Pen.stroke;
Pen.moveTo(5@100);
Pen.lineTo(10@100);
Pen.stroke;
Pen.moveTo(5@110);
Pen.lineTo(10@110);
Pen.stroke;
Pen.moveTo(5@120);
Pen.lineTo(10@120);
Pen.stroke;
Pen.moveTo(5@130);
Pen.lineTo(10@130);
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
Pen.stroke;
Pen.moveTo(30@210);
Pen.lineTo(30@215);
Pen.stroke;
Pen.moveTo(40@210);
Pen.lineTo(40@215);
```

```
        Pen.stroke;
        Pen.moveTo(50@210);
        Pen.lineTo(50@215);
        Pen.stroke;
        Pen.moveTo(60@210);
        Pen.lineTo(60@215);
        Pen.stroke;
        Pen.moveTo(70@210);
        Pen.lineTo(70@215);
        Pen.stroke;
        Pen.moveTo(80@210);
        Pen.lineTo(80@215);
        Pen.stroke;
        Pen.moveTo(90@210);
        Pen.lineTo(90@215);
        Pen.stroke;
        Pen.moveTo(100@210);
        Pen.lineTo(100@215);
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@200);
Pen.lineTo(130@202);
Pen.lineTo(130@43.8);
Pen.lineTo(10@50.2);
        Pen.fill;

        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@197.8);
        Pen.lineTo(70@200);
        Pen.stroke;
        Pen.moveTo(70@134.9);
        Pen.lineTo(130@135.7);
        Pen.stroke;
        Pen.moveTo(70@148.3);
        Pen.lineTo(130@148.8);
        Pen.stroke;
        Pen.moveTo(70@172.3);
        Pen.lineTo(130@173.5);
        Pen.stroke;
        Pen.moveTo(70@187.8);
        Pen.lineTo(130@189.4);
        Pen.stroke;
```

```
        Pen.fillColor = Color.blue(1, 0.9);
    Pen.moveTo(10@10);

    Pen.lineTo(30@10);
    Pen.lineTo(30@20);
    Pen.lineTo(10@20);
    Pen.lineTo(10@10);
        Pen.fill;

        Pen.fillColor = Color.red(1, 0.9);
    Pen.moveTo(61.22@10);
    Pen.lineTo(78.36@10);
    Pen.lineTo(78.36@20);
    Pen.lineTo(61.22@20);
    Pen.lineTo(61.22@10);
        Pen.fill;

        Pen.moveTo(101.1@10);
    Pen.lineTo(126.54@10);
    Pen.lineTo(126.54@20);
    Pen.lineTo(101.1@20);
    Pen.lineTo(101.1@10);
        Pen.fill;

        Pen.fillColor = Color.gray(0.1, 0.4);
        Pen.font = Font("Birch Std", 45);
        Pen.stringAtPoint("22", (100@40));

//button 23

        Pen.translate(157,0);
        Pen.strokeColor = Color.black;

        Pen.moveTo(0@0);
        Pen.lineTo(140@0);
        Pen.lineTo(140@220);
        Pen.lineTo(0@220);
        Pen.lineTo(0@0);
        Pen.stroke;

        Pen.moveTo(10@10);
    Pen.lineTo(10@210);
        Pen.lineTo(130@210);
    Pen.stroke;

        Pen.moveTo(5@10);
        Pen.lineTo(10@10);
        Pen.stroke;
        Pen.moveTo(5@20);
        Pen.lineTo(10@20);
        Pen.stroke;
        Pen.moveTo(5@30);
        Pen.lineTo(10@30);
```

```
Pen.stroke;
Pen.moveTo(5@40);
Pen.lineTo(10@40);
Pen.stroke;
Pen.moveTo(5@50);
Pen.lineTo(10@50);
Pen.stroke;
Pen.moveTo(5@60);
Pen.lineTo(10@60);
Pen.stroke;
Pen.moveTo(5@70);
Pen.lineTo(10@70);
Pen.stroke;
Pen.moveTo(5@80);
Pen.lineTo(10@80);
Pen.stroke;
Pen.moveTo(5@90);
Pen.lineTo(10@90);
Pen.stroke;
Pen.moveTo(5@100);
Pen.lineTo(10@100);
Pen.stroke;
Pen.moveTo(5@110);
Pen.lineTo(10@110);
Pen.stroke;
Pen.moveTo(5@120);
Pen.lineTo(10@120);
Pen.stroke;
Pen.moveTo(5@130);
Pen.lineTo(10@130);
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
```

```
        Pen.stroke;
        Pen.moveTo(30@210);
        Pen.lineTo(30@215);
        Pen.stroke;
        Pen.moveTo(40@210);
        Pen.lineTo(40@215);
        Pen.stroke;
        Pen.moveTo(50@210);
        Pen.lineTo(50@215);
        Pen.stroke;
        Pen.moveTo(60@210);
        Pen.lineTo(60@215);
        Pen.stroke;
        Pen.moveTo(70@210);
        Pen.lineTo(70@215);
        Pen.stroke;
        Pen.moveTo(80@210);
        Pen.lineTo(80@215);
        Pen.stroke;
        Pen.moveTo(90@210);
        Pen.lineTo(90@215);
        Pen.stroke;
        Pen.moveTo(100@210);
        Pen.lineTo(100@215);
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@202);
        Pen.lineTo(26@195.6);
        Pen.lineTo(44.4@181.8);
        Pen.lineTo(88.4@201.5);
        Pen.lineTo(108.4@200.5);
        Pen.lineTo(114.4@186.9);
        Pen.lineTo(122.4@185.9);
Pen.lineTo(130@203.7);
Pen.lineTo(130@37.2);
        Pen.lineTo(124.4@76.2);
        Pen.lineTo(104.4@66.3);
        Pen.lineTo(94.4@80.5);
        Pen.lineTo(64.4@28.3);
        Pen.lineTo(44.4@93.7);
        Pen.lineTo(24.4@36.8);
Pen.lineTo(10@43.8);
        Pen.fill;
```

```
        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@133.3);
        Pen.lineTo(70@134.4);
        Pen.stroke;
        Pen.moveTo(70@161.8);
        Pen.lineTo(130@162.6);
        Pen.stroke;
        Pen.moveTo(10@148.2);
        Pen.lineTo(70@148.7);
        Pen.stroke;
        Pen.moveTo(10@171.2);
        Pen.lineTo(70@172.1);
        Pen.stroke;
        Pen.moveTo(70@181.5);
        Pen.lineTo(130@182.7);
        Pen.stroke;
        Pen.moveTo(10@188.5);
        Pen.lineTo(70@191.1);
        Pen.stroke;


        Pen.fillColor = Color.blue(1, 0.9);
    Pen.moveTo(10@10);

    Pen.lineTo(16@10);
    Pen.lineTo(16@20);
    Pen.lineTo(10@20);
    Pen.lineTo(10@10);
        Pen.fill;

        Pen.fillColor = Color.red(1, 0.9);
    Pen.moveTo(47.14@10);
    Pen.lineTo(74.28@10);
    Pen.lineTo(74.28@20);
    Pen.lineTo(47.14@20);
    Pen.lineTo(47.14@10);
        Pen.fill;

        Pen.moveTo(106@10);
    Pen.lineTo(132@10);
    Pen.lineTo(132@20);
    Pen.lineTo(106@20);
    Pen.lineTo(106@10);
        Pen.fill;

        Pen.fillColor = Color.gray(0.1, 0.4);
        Pen.font = Font("Birch Std", 45);
        Pen.stringAtPoint("23", (100@40));

//button 24

        Pen.translate(157,0);
        Pen.strokeColor = Color.black;
```

```
        Pen.moveTo(0@0);
        Pen.lineTo(140@0);
        Pen.lineTo(140@220);
        Pen.lineTo(0@220);
        Pen.lineTo(0@0);
        Pen.stroke;

        Pen.moveTo(10@10);
Pen.lineTo(10@210);
        Pen.lineTo(130@210);
Pen.stroke;

        Pen.moveTo(5@10);
        Pen.lineTo(10@10);
        Pen.stroke;
        Pen.moveTo(5@20);
        Pen.lineTo(10@20);
        Pen.stroke;
        Pen.moveTo(5@30);
        Pen.lineTo(10@30);
        Pen.stroke;
        Pen.moveTo(5@40);
        Pen.lineTo(10@40);
        Pen.stroke;
        Pen.moveTo(5@50);
        Pen.lineTo(10@50);
        Pen.stroke;
        Pen.moveTo(5@60);
        Pen.lineTo(10@60);
        Pen.stroke;
        Pen.moveTo(5@70);
        Pen.lineTo(10@70);
        Pen.stroke;
        Pen.moveTo(5@80);
        Pen.lineTo(10@80);
        Pen.stroke;
        Pen.moveTo(5@90);
        Pen.lineTo(10@90);
        Pen.stroke;
        Pen.moveTo(5@100);
        Pen.lineTo(10@100);
        Pen.stroke;
        Pen.moveTo(5@110);
        Pen.lineTo(10@110);
        Pen.stroke;
        Pen.moveTo(5@120);
        Pen.lineTo(10@120);
        Pen.stroke;
        Pen.moveTo(5@130);
        Pen.lineTo(10@130);
        Pen.stroke;
        Pen.moveTo(5@140);
        Pen.lineTo(10@140);
        Pen.stroke;
```

```
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
Pen.stroke;
Pen.moveTo(30@210);
Pen.lineTo(30@215);
Pen.stroke;
Pen.moveTo(40@210);
Pen.lineTo(40@215);
Pen.stroke;
Pen.moveTo(50@210);
Pen.lineTo(50@215);
Pen.stroke;
Pen.moveTo(60@210);
Pen.lineTo(60@215);
Pen.stroke;
Pen.moveTo(70@210);
Pen.lineTo(70@215);
Pen.stroke;
Pen.moveTo(80@210);
Pen.lineTo(80@215);
Pen.stroke;
Pen.moveTo(90@210);
Pen.lineTo(90@215);
Pen.stroke;
Pen.moveTo(100@210);
Pen.lineTo(100@215);
Pen.stroke;
Pen.moveTo(110@210);
Pen.lineTo(110@215);
Pen.stroke;
Pen.moveTo(120@210);
Pen.lineTo(120@215);
Pen.stroke;
Pen.moveTo(130@210);
Pen.lineTo(130@215);
Pen.stroke;
```

```
        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@203.7);
        Pen.lineTo(22.4@200.5);
        Pen.lineTo(36.4@178.2);
        Pen.lineTo(82.4@182.4);
        Pen.lineTo(92.4@69.5);
        Pen.lineTo(100.4@80.4);
        Pen.lineTo(112.4@191.7);
        Pen.lineTo(122@180.1);
Pen.lineTo(130@205.2);
Pen.lineTo(130@30.8);
        Pen.lineTo(104.4@36.8);
        Pen.lineTo(82.4@62.3);
        Pen.lineTo(72.4@50.5);
Pen.lineTo(60.4@101.4);
        Pen.lineTo(32@41.3);
        Pen.lineTo(10@37.2);
        Pen.fill;

        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@140.7);
        Pen.lineTo(70@141.5);
        Pen.stroke;
        Pen.moveTo(70@188.6);
        Pen.lineTo(130@189.7);
        Pen.stroke;
        Pen.moveTo(10@149.7);
        Pen.lineTo(70@150.3);
        Pen.stroke;
        Pen.moveTo(10@175.2);
        Pen.lineTo(70@176);
        Pen.stroke;
        Pen.moveTo(10@184.8);
        Pen.lineTo(70@183.5);
        Pen.stroke;

        Pen.fillColor = Color.blue(1, 0.9);
Pen.moveTo(10@10);

Pen.lineTo(30@10);
Pen.lineTo(30@20);
Pen.lineTo(10@20);
Pen.lineTo(10@10);
        Pen.fill;

        Pen.fillColor = Color.red(1, 0.9);
Pen.moveTo(30@10);
Pen.lineTo(60@10);
Pen.lineTo(60@20);
Pen.lineTo(30@20);
Pen.lineTo(30@10);
```

```
            Pen.fill;

            Pen.moveTo(108.04@10);
    Pen.lineTo(130.62@10);
    Pen.lineTo(130.62@20);
    Pen.lineTo(108.04@20);
    Pen.lineTo(108.04@10);
            Pen.fill;

            Pen.fillColor = Color.gray(0.1, 0.4);
            Pen.font = Font("Birch Std", 45);
            Pen.stringAtPoint("24", (100@40));

//button 25

            Pen.translate(157,0);
            Pen.strokeColor = Color.black;

            Pen.moveTo(0@0);
            Pen.lineTo(140@0);
            Pen.lineTo(140@220);
            Pen.lineTo(0@220);
            Pen.lineTo(0@0);
            Pen.stroke;

            Pen.moveTo(10@10);
    Pen.lineTo(10@210);
            Pen.lineTo(130@210);
    Pen.stroke;

            Pen.moveTo(5@10);
            Pen.lineTo(10@10);
            Pen.stroke;
            Pen.moveTo(5@20);
            Pen.lineTo(10@20);
            Pen.stroke;
            Pen.moveTo(5@30);
            Pen.lineTo(10@30);
            Pen.stroke;
            Pen.moveTo(5@40);
            Pen.lineTo(10@40);
            Pen.stroke;
            Pen.moveTo(5@50);
            Pen.lineTo(10@50);
            Pen.stroke;
            Pen.moveTo(5@60);
            Pen.lineTo(10@60);
            Pen.stroke;
            Pen.moveTo(5@70);
            Pen.lineTo(10@70);
            Pen.stroke;
            Pen.moveTo(5@80);
            Pen.lineTo(10@80);
            Pen.stroke;
            Pen.moveTo(5@90);
```

```
Pen.lineTo(10@90);
Pen.stroke;
Pen.moveTo(5@100);
Pen.lineTo(10@100);
Pen.stroke;
Pen.moveTo(5@110);
Pen.lineTo(10@110);
Pen.stroke;
Pen.moveTo(5@120);
Pen.lineTo(10@120);
Pen.stroke;
Pen.moveTo(5@130);
Pen.lineTo(10@130);
Pen.stroke;
Pen.moveTo(5@140);
Pen.lineTo(10@140);
Pen.stroke;
Pen.moveTo(5@150);
Pen.lineTo(10@150);
Pen.stroke;
Pen.moveTo(5@160);
Pen.lineTo(10@160);
Pen.stroke;
Pen.moveTo(5@170);
Pen.lineTo(10@170);
Pen.stroke;
Pen.moveTo(5@180);
Pen.lineTo(10@180);
Pen.stroke;
Pen.moveTo(5@190);
Pen.lineTo(10@190);
Pen.stroke;
Pen.moveTo(5@200);
Pen.lineTo(10@200);
Pen.stroke;

Pen.moveTo(20@210);
Pen.lineTo(20@215);
Pen.stroke;
Pen.moveTo(30@210);
Pen.lineTo(30@215);
Pen.stroke;
Pen.moveTo(40@210);
Pen.lineTo(40@215);
Pen.stroke;
Pen.moveTo(50@210);
Pen.lineTo(50@215);
Pen.stroke;
Pen.moveTo(60@210);
Pen.lineTo(60@215);
Pen.stroke;
Pen.moveTo(70@210);
Pen.lineTo(70@215);
Pen.stroke;
Pen.moveTo(80@210);
```

```
        Pen.lineTo(80@215);
        Pen.stroke;
        Pen.moveTo(90@210);
        Pen.lineTo(90@215);
        Pen.stroke;
        Pen.moveTo(100@210);
        Pen.lineTo(100@215);
        Pen.stroke;
        Pen.moveTo(110@210);
        Pen.lineTo(110@215);
        Pen.stroke;
        Pen.moveTo(120@210);
        Pen.lineTo(120@215);
        Pen.stroke;
        Pen.moveTo(130@210);
        Pen.lineTo(130@215);
        Pen.stroke;

        Pen.fillColor = Color.green(0.5, 0.9);
Pen.moveTo(10@107);

Pen.lineTo(10@205.2);
        Pen.lineTo(26.4@204.8);
        Pen.lineTo(44.4@205.3);
        Pen.lineTo(62.4@184.9);
        Pen.lineTo(88.4@200);
        Pen.lineTo(100.4@205.2);
        Pen.lineTo(106.4@193);
        Pen.lineTo(120.4@200.9);
Pen.lineTo(130@205.9);
Pen.lineTo(130@18.2);
        Pen.lineTo(120.4@94.3);
        Pen.lineTo(98.4@67.2);
        Pen.lineTo(92.4@150.4);
        Pen.lineTo(80.4@41.5);
        Pen.lineTo(66.4@101.9);
        Pen.lineTo(48.4@50.8);
        Pen.lineTo(30.4@92.3);
Pen.lineTo(10@30.8);
        Pen.fill;

        Pen.strokeColor = Color.magenta(0.3, 0.9);
        Pen.width = 3;
        Pen.moveTo(10@165.2);
        Pen.lineTo(70@166.7);
        Pen.stroke;
        Pen.moveTo(70@179.9);
        Pen.lineTo(130@180.7);
        Pen.stroke;
        Pen.moveTo(10@181.5);
        Pen.lineTo(70@181.9);
        Pen.stroke;
        Pen.moveTo(70@182.5);
        Pen.lineTo(130@182.9);
        Pen.stroke;
```

```
		Pen.fillColor = Color.blue(1, 0.9);
	Pen.moveTo(10@10);

	Pen.lineTo(16@10);
	Pen.lineTo(16@20);
	Pen.lineTo(10@20);
	Pen.lineTo(10@10);
		Pen.fill;

		Pen.fillColor = Color.red(1, 0.9);
	Pen.moveTo(49.14@10);
	Pen.lineTo(78.3@10);
	Pen.lineTo(78.3@20);
	Pen.lineTo(49.14@20);
	Pen.lineTo(49.14@10);
		Pen.fill;

		Pen.moveTo(101.72@10);
	Pen.lineTo(123.44@10);
	Pen.lineTo(123.44@20);
	Pen.lineTo(101.72@20);
	Pen.lineTo(101.72@10);
		Pen.fill;

		Pen.fillColor = Color.gray(0.1, 0.4);
		Pen.font = Font("Birch Std", 45);
		Pen.stringAtPoint("25", (100@40));

// shift
		Pen.scale(1.3, 1.3);
		Pen.translate(160,-360);

// legend

		Pen.strokeColor = Color.black;
		Pen.fillColor = Color.black;
	Pen.moveTo(80@80);
	Pen.lineTo(80@480);
		Pen.lineTo(320@480);
	Pen.stroke;

		Pen.font = Font("Arial", 30);
		Pen.stringAtPoint("Button Guide", (5@5));

		Pen.moveTo(75@80);
		Pen.lineTo(80@80);
		Pen.stroke;
		Pen.moveTo(75@100);
		Pen.lineTo(80@100);
		Pen.stroke;
		Pen.moveTo(75@120);
		Pen.lineTo(80@120);
		Pen.stroke;
```

```
Pen.moveTo(75@140);
Pen.lineTo(80@140);
Pen.stroke;
Pen.moveTo(75@160);
Pen.lineTo(80@160);
Pen.stroke;
Pen.moveTo(75@180);
Pen.lineTo(80@180);
Pen.stroke;
Pen.moveTo(75@200);
Pen.lineTo(80@200);
Pen.stroke;
Pen.moveTo(75@220);
Pen.lineTo(80@220);
Pen.stroke;
Pen.moveTo(75@240);
Pen.lineTo(80@240);
Pen.stroke;
Pen.moveTo(75@260);
Pen.lineTo(80@260);
Pen.stroke;
Pen.moveTo(75@280);
Pen.lineTo(80@280);
Pen.stroke;
Pen.moveTo(75@300);
Pen.lineTo(80@300);
Pen.stroke;
Pen.moveTo(75@320);
Pen.lineTo(80@320);
Pen.stroke;
Pen.moveTo(75@340);
Pen.lineTo(80@340);
Pen.stroke;
Pen.moveTo(75@360);
Pen.lineTo(80@360);
Pen.stroke;
Pen.moveTo(75@380);
Pen.lineTo(80@380);
Pen.stroke;
Pen.moveTo(75@400);
Pen.lineTo(80@400);
Pen.stroke;
Pen.moveTo(75@420);
Pen.lineTo(80@420);
Pen.stroke;
Pen.moveTo(75@440);
Pen.lineTo(80@440);
Pen.stroke;
Pen.moveTo(75@460);
Pen.lineTo(80@460);
Pen.stroke;

Pen.font = Font("Arial", 9);
Pen.stringAtPoint("60", (59@456));
Pen.stringAtPoint("240", (54@436));
```

```
        Pen.stringAtPoint("540", (54@416));
        Pen.stringAtPoint("960", (54@396));
        Pen.stringAtPoint("1500", (50@376));
        Pen.stringAtPoint("2160", (50@356));
        Pen.stringAtPoint("2940", (50@336));
        Pen.stringAtPoint("3840", (50@316));
        Pen.stringAtPoint("4860", (50@296));
        Pen.stringAtPoint("6000", (50@276));
        Pen.stringAtPoint("7260", (50@256));
        Pen.stringAtPoint("8640", (50@236));
        Pen.stringAtPoint("10140", (46@216));
        Pen.stringAtPoint("11760", (46@196));
        Pen.stringAtPoint("13500", (46@176));
        Pen.stringAtPoint("15360", (46@156));
        Pen.stringAtPoint("17340", (46@136));
        Pen.stringAtPoint("19440", (46@116));
        Pen.stringAtPoint("21660", (46@96));
        Pen.stringAtPoint("24000", (46@76));
        Pen.font = Font("Arial", 12);
        Pen.stringAtPoint("Freq.", (2@230));
        Pen.stringAtPoint("in Hertz", (2@250));


        Pen.moveTo(120@480);
        Pen.lineTo(120@485);
        Pen.stroke;
        Pen.moveTo(160@480);
        Pen.lineTo(160@485);
        Pen.stroke;
        Pen.moveTo(200@480);
        Pen.lineTo(200@485);
        Pen.stroke;
        Pen.moveTo(240@480);
        Pen.lineTo(240@485);
        Pen.stroke;
        Pen.moveTo(280@480);
        Pen.lineTo(280@485);
        Pen.stroke;
        Pen.moveTo(320@480);
        Pen.lineTo(320@485);
        Pen.stroke;

        Pen.font = Font("Arial", 9);
        Pen.stringAtPoint("10", (115@487));
        Pen.stringAtPoint("20", (155@487));
        Pen.stringAtPoint("30", (195@487));
        Pen.stringAtPoint("40", (235@487));
        Pen.stringAtPoint("50", (275@487));
        Pen.stringAtPoint("60", (315@487));
        Pen.font = Font("Arial", 12);
        Pen.stringAtPoint("Time in Seconds", (160@500));


        Pen.fillColor = Color.green(0.7, 0.6);
Pen.moveTo(80@316);
```

```
Pen.lineTo(315@415);
Pen.lineTo(315@147);
	Pen.lineTo(80@267);
	Pen.fill;
	Pen.fillColor = Color.black;
	Pen.font = Font("Arial", 11);
	Pen.stringAtPoint("freq. parameters of ", (190@235));
	Pen.stringAtPoint("band-pass filter ", (190@250));
	Pen.stringAtPoint("on live sound in ", (190@265));

	Pen.strokeColor = Color.magenta(0.9, 0.6);
	Pen.width = 3;
	Pen.moveTo(80@342.8);
	Pen.lineTo(195@345.2);
	Pen.stroke;
	Pen.moveTo(195@366.3);
	Pen.lineTo(315@366.6);
	Pen.stroke;
	Pen.moveTo(80@350);
	Pen.lineTo(195@350.5);
	Pen.stroke;
	Pen.moveTo(80@376.2);
	Pen.lineTo(195@377.3);
	Pen.stroke;
	Pen.moveTo(195@381.8);
	Pen.lineTo(315@382.2);
	Pen.stroke;
	Pen.moveTo(80@382.8);
	Pen.lineTo(195@383);
	Pen.stroke;
	Pen.font = Font("Arial", 11);
	Pen.stringAtPoint("freq. of sine wave trajectories ", (100@385));
	Pen.stringAtPoint("modulating live sound in ", (100@397));


	Pen.fillColor = Color.blue(1, 0.6);
Pen.moveTo(90@55);

Pen.lineTo(146@55);
Pen.lineTo(146@75);
Pen.lineTo(90@75);
Pen.lineTo(90@55);
	Pen.fill;
	Pen.fillColor = Color.black;
	Pen.font = Font("Arial", 11);
	Pen.stringAtPoint("record sound in", (86@76));
Pen.stringAtPoint("over this duration", (86@90));

	Pen.fillColor = Color.red(1, 0.6);
Pen.moveTo(180@55);
Pen.lineTo(250@55);
Pen.lineTo(250@75);
Pen.lineTo(180@75);
Pen.lineTo(180@55);
	Pen.fill;
```

```
            Pen.moveTo(275@55);
    Pen.lineTo(315@55);
    Pen.lineTo(315@75);
    Pen.lineTo(275@75);
    Pen.lineTo(275@55);
            Pen.fill;
            Pen.fillColor = Color.black;
            Pen.font = Font("Arial", 11);
            Pen.stringAtPoint("playback events", (210@76));
            Pen.stringAtPoint("at varying rates", (210@90));



};


w.front;
);
);

} // wait for boot
```